Data mining by means of generalized patterns

POLITECNICO DI TORINO

PhD in Information and System Engineering
XXIV cycle

III Facoltà di Ingegneria
Settore scientifico ING-INF/05

PhD Thesis

Data mining by means of generalized patterns

Author:
Luca Cagliero
Matr. 160399

Supervisor:
Prof. Elena Baralis

March 2012
A.A. 2011/2012

Dedicated to Giulia and to my family

Acknowledgments

I would like thank Professor Elena Baralis for her insightful suggestions and comments. Her advices were precious and fundamental for the development of my PhD work. I will grateful to her for providing me the possibility of getting involved in very interesting projects.

My PhD and my academic career probably would not have been possible without the help of the people that accompanied me during the last three years. I want to thank all my research group colleagues, Paolo, Tania, Daniele, Alessandro, Giulia, Alberto, and Vincenzo for their constant support and suggestions.

A special thank goes to Giulia and to my family for their constant belief and support.

Contents

Chapter 1

Introduction

The data mining and knowledge discovery (KDD) process accomplishes the task of converting the raw data into useful information [107]. The core part of data mining process entails the application of predictive or descriptive techniques to analyze the data of interest. A widely used descriptive data mining technique is pattern discovery (also called association analysis). It focuses on discovering associations (i.e., the patterns) that represent strong correlations among data [107]. Useful applications of this analysis in different real-life contexts have been already proposed in literature (e.g., medical images [7], network traffic analysis [15], biological data [37], context-aware applications [50, 112]).

The suitability of pattern discovery approaches for effectively supporting business analysis and decision making strictly depends on the granularity level of the available information. This thesis is mainly focused on the study and the application of pattern discovery algorithms that aggregate database knowledge to discover and exploit valuable correlations, hidden in the analyzed data, at different abstraction levels. More specifically, the aim of the research effort described in this work is two-fold: the discovery of associations, in the form of generalized patterns, from large data collections and the inference of semantic models, i.e., taxonomies and ontologies, suitable for driving the mining process.

Association discovery focuses on discovering hidden correlations among data. Taxonomies provide hierarchies of aggregations over the analyzed data that may be exploited to drive the extraction of correlations at different abstraction levels, i.e., the generalized patterns. The automatic inference of sound semantic models, such as taxonomies and ontologies, customized on the analyzed data may be effectively combined with the process of generalized

pattern discovery to ease the domain expert's task and, thus, to enhance the knowledge discovery process. To this aim, a semi-automatic approach [32] to support the construction of ontologies and taxonomies based on well-founded database and data mining techniques, i.e., functional dependency discovery [107] and association rule discovery [3], is presented.

The study of novel and more effective generalized pattern mining algorithms is focused on selecting the most informative and not redundant patterns and making the extracted pattern sets manageable by domain experts. In the context of frequent itemset and association rule mining [3], a significant research effort has been devoted to (i) preventing the generation of redundant or less interesting patterns by pushing ad-hoc constraints into the mining process and (ii) selecting most valuable patterns by means of postprocessing steps. This thesis addresses both issues in the context of generalized itemsets and rules [103]. It presents an effective generalized itemset mining algorithm [16] that prevents the extraction of redundant patterns by lazily evaluating the given taxonomy and a rule mining constraint useful for reducing the number of generated high level rules [17].

A parallel effort has been devoted to perform change mining from timestamped data collections by means of association discovery. Change mining concerns the analysis and the comparison of the information extracted by different data mining and knowledge discovery sessions scheduled over time. The aim is to track the evolution of patterns over consecutive time periods to discover and select most significant history patterns, i.e., patterns that represent the evolution of correlations among data from one time period to another. This thesis also addresses the usage of generalized pattern in change mining. More specifically, a type of history pattern, i.e., the history generalized pattern [31], that represents dynamic knowledge at different abstraction levels is presented. The history generalized patterns may include generalizations of the same low level pattern to prevent the discarding of possibly relevant knowledge that unexpectedly occurs rarely in a certain time period.

Finally, the application of the previously described mining patterns and algorithms [16, 17, 31, 32] to a number of real application contexts is discussed. Experimental results achieved on real data coming from mobile context-aware applications [19], the network traffic domain [16, 17], and social networks analysis [33, 34] show the effectiveness of the proposed approaches. The obtained results have been validated by domain experts to select most relevant patterns and figure out most useful applications for each domain.

This thesis is organized as follows. Chapter 2 overviews main research efforts pertaining to generalized pattern discovery, change mining, seman-

tic model construction, and their applications in real application contexts. Chapter 3 presents the background knowledge about generalized pattern discovery and formally states the generalized itemset and association rule mining problems. Chapter 4 presents an algorithm, namely GenIO (GENeralized Itemset DiscOverer), to accomplish the generalized itemset mining task effectively and efficiently. Furthermore, it also presents three data mining systems based on the same algorithm to perform knowledge discovery from data coming from different application domains. Chapter 5 presents constraints to push into the generalized association rule mining process to select the patterns of interests. Chapter 6 addresses the problem of change mining in the presence of taxonomies by introducing the HiGen (History GENeralized Pattern), while Chapter 7 presents a semi-automatic approach for semantic model construction. Finally, Chapter 8 draws conclusions and presents future developments for the discussed approaches.

Chapter 2

Related work

This chapter overviews main research work pertaining to the research problems tackled by this thesis. In particular, Section 2.1 present most relevant work concerning generalized pattern discovery. Sections 2.2 and 2.3 presents most significant research efforts concerning, respectively, pattern change mining and the inference of meaningful semantic models (e.g., ontologies and taxonomies) customized on the analyzed data. Finally, Section 2.4 overviews main data mining applications contexts in which the use of generalized patterns has been already investigated, i.e., context-aware systems, network traffic analysis, and social network analysis.

2.1 Generalized pattern discovery

A significant research effort has been devoted to the design and the development of novel algorithms to efficiently extract generalized itemsets and association rules. This issue was first addressed in [103] to perform market basket analysis. The first generalized association rule mining algorithm [103] generates itemsets by considering, for each item, all its parents in a taxonomy (i.e., a is-a hierarchy built over data items). Hence, candidate frequent itemsets are generated by exhaustively evaluating the taxonomy and, thus, by extracting a large amount of redundant patterns.

One step further towards a more efficient extraction process for generalized association rules was based on new optimization strategies [16, 52, 58, 94]. In [58] a faster support counting is proposed to compute the TID intersection in algorithms that exploit the vertical data format [122]. Differently, in [52], an optimization strategy based on a top-down hierarchy traversal

is proposed. It identifies in advance itemsets that cannot be frequent in a transactional dataset by exploiting the Apriori principle [2]. To further prune the search space, it also proposes to mainly focus on a valuable subset of generalized itemsets, namely the level-sharing itemset, characterized by items belonging to the same level of the taxonomy. In [52] the discovery of interesting generalized association rules is driven by the enforcement of a level-dependent multiple support threshold when level-sharing itemsets are extracted. The idea to drive the extraction of traditional generalized itemsets by means of opportunistic aggregation has been first proposed in [16] and exploited in [8, 17, 19, 31, 34]. The algorithm proposed in [16], namely GenIO (GENeralized Itemset DiscOverer), discovers all frequent itemsets and all (traditional) generalized itemsets having at least an infrequent descendant. It has been profitability exploited in different application domains (i.e., network traffic analysis [8], context-aware application [19], social network mining [34]) where the extracted knowledge has deemed valuable by the corresponding domain experts. A more through description of the approaches proposed in [17, 19, 31, 34] is given in the following sections.

Typically, the analyst is not interested in all the frequent (generalized) itemsets or rules. Hence, many previous works [5, 9, 23, 105, 116] have been devoted to enforcing constraints to extract only a subset of the patterns of interest. Some of them are based on the items of interest according to the analyst preferences (e.g., [5, 9, 105]) while others are based on statistical and objective measures (e.g., [23, 116]). Since the analyst commonly knows the items or the type of patterns he/she is mainly interested in, this knowledge can be used to prune the search space. The first algorithm that allows the user to specify a set of constraints on the patterns of interest is proposed in [5]. It allows specifying a set of item constraints, which are used to select the items of interest and how they could be combined, by means of boolean expressions (e.g., the presence and the absence of items into the mined set). Differently, in [105], subset-superset and parent-child relationships in the lattice of generalized patterns are exploited to constrain the mining process. Futhermore, authors in [12] propose to extend the traditional association rule mining problem [2] by exploiting a broader set of logical operators that could appear in data items, rather than just the equality. This approach allows analysts to state the item value ranges of interest. However, the usage of both item constraints [5] and more general item forms [12] requires to explicate all the items that can appear together and those that cannot. Hence, constraints based on the pattern schema are definitely more compact and easy to use. In [9] an ad-hoc language to enforce constraints on the characteristics of the rule body and head has been proposed. Other approaches (e.g., [116]) exploit

objective measures and statistical tests to perform pattern selection, instead of analyst preferences. For example, in [116] a set of statistical tests have been used to select significant patterns. The proposed approach is orthogonal with respect to the aforementioned ones [5, 9] as it could be applied to select the most statistically significant patterns among those of analyst's interest. Finally, authors in [84] combine the usage of objective measures with a subjective association rule evaluation driven by ontologies to select the subset of rules of interest. Authors in [17] exploit schema constraints, i.e., constraints at the level of the database schema, to early patterns not of analyst's interest during the generalized association rule mining process. Furthermore, a constraint on the set of extracted high level rules, namely the generalized confidence constraint, is proposed with the aim at preventing the extraction of redundant high level rules. A more through description of the approach proposed in [17] is given in Chapter 5.

In the context of fuzzy association rules [71] the problem of generalized rule mining has been also addressed. In [73] a top-down deepening approach has been adopted in fuzzy association rule mining by enforcing, as in [52, 79], different minimum support thresholds on different items.

A parallel effort has been devoted to mining generalized rules from both categorical and quantitative (i.e., numerical) data [4], by extending the concFrequent ept of boolean association rules introduced in [2]. While categorical attributes are aggregated by following a user-provided taxonomy, quantitative data are aggregated by minimizing the information loss measure [53].

2.2 Pattern change mining

Frequent itemset mining has been first proposed in [2], in the context of market basket analysis, as the first step of the association rule extraction process. The problem of discovering relevant changes in the history of itemsets and association rules has been already addressed by a number of research papers (e.g., [6, 10, 21, 26, 44, 80]). Active data mining [6] was the first attempt to represent and query the history pattern of the discovered association rule quality indexes. It first mines rules, from datasets collected in different time periods, by adding rules and their related quality indexes (e.g., support and confidence [2]) to a common rule base. Next, the analyst could specify a history pattern in a trigger which is fired when such a pattern trend is exhibiting. More recently, other time-related data mining frameworks tailored to monitor and detect changes in rule support and confidence have been pro-

posed [21, 26, 109]. In [26] patterns are evaluated and pruned based on both subjective and objective interestingness measures. The aim of [21] is to monitor the mined patterns and to reduce the effort spent in data mining. To achieve this goal, new patterns are observed as soon as they emerge, and old patterns are removed from the rule base as soon as they become extinct. To further reduce the computational cost, at one time period a subset of rules is selected and monitored, while data changes that occur in subsequent periods are measured by their impact on the rules being monitored. Similarly, in [109] itemset change mining from time-varying data streams is addressed. Differently,the work presented in [82] deals with rule change mining by discovering two main types of rules: (i) the stable rules, i.e., rules that do not change a great deal over time and, thus, are more reliable and could be trusted and (ii) the trend rules, i.e., rules that indicate some underlying systematic trends of potential interest.

The problem of discovering the subset of most relevant changes in association rules has been addressed by (i) evaluating rule changes that occur between two time periods by means of chi-square test [80], (ii) searching for border descriptions of emerging patterns extracted from a pair of datasets [44], and (iii) applying a fuzzy approach to rule change evaluation [10]. Unlike [10, 44, 80], the approach proposed in [31] addresses change mining in the presence of taxonomies by representing patterns that become infrequent in a certain time period by means of one of their generalizations characterized by minimal redundancy. A more detailed description of the abovementioned approach is reported in Chapter 6.

A parallel issue concerns the detection of most notable changes in multidimensional data measures. Authors in [65] first introduce the concept of cubegrade and compare the multidimensional data cube cells with their gradient cells, namely their ancestors, descendants, and siblings. In [43] authors extend the work proposed in [65] by pushing anti-monotone constraints into the mining of highly similar data cube cell pairs with hugely different measure values.

2.3 inference of semantic models

An ontology is a complete domain knowledge representation through concepts and the corresponding relationships. Semantic Web tools provide the platforms to significantly enrich knowledge representation through a wide range of semantic-based models. These models support users in understand-

ing the meaning of a resource and the related domain. However, the creation of ontologies in a Web Ontology Language, like OWL [115], heavily rely on human intervention. Thus, the automatic construction of conceptual ontologies is becoming an increasingly appealing target in several research fields, including information retrieval, data mining, data summarization, and text categorization.

Ontology inference has been addressed in different application contexts, such as social network analysis [51] and e-learning environments [72]. Offline ontology generation addresses the construction of hierarchical data models that could be effectively and efficiently used by the online systems. In [51] the authors proposed the integration of social relations coming from social networks with collaborative tagging to extract lightweight ontologies by means of hierarchical clustering. Differently, in [72], the extraction of ontologies from textual messages in an e-learning environment is proposed. To this aim, it performs fuzzy text mining and categorization by exploiting statistical measures (e.g., mutual information) to evaluate term significance. Inferred ontologies were posted on online forums to support instructors in quickly identifying the progress of their students.

A parallel effort has been devoted to either constructing or automatically building a taxonomy, which is a specialization of an ontology. A taxonomy is a hierarchical organization of concepts, topics, and keywords in which only is-a relationships among concepts hold. Several works address taxonomy construction by exploiting well-known data mining techniques. Mostly proposed algorithms exploited clustering techniques to provide a well-founded structuring of concepts of interest [36, 42, 49, 64]. The mostly used techniques are based on hierarchical clustering, which produces a set of nested clusters organized as a hierarchical tree, called dendrogram. The most relevant application context in which clustering techniques have been adopted to address taxonomy construction is the context of textual data analysis [36, 42, 49, 54, 59, 64]. A similar technique has been also applied to the results of Web search engines to improve user browsing [123] and to enhance the quality of recommendation systems [77].

To overcome well-known hierarchical clustering algorithm limitations in terms of complexity and optimality, diverse techniques have been devised in different application domains [86, 97, 117]. In the context of textual data analysis, in [86] authors first proposed to exploit neural networks to produce a compound similarity measure of words. Differently, in [117] salient words are extracted from documents and hierarchically organized by using, similarly to [97], term co-occurrence frequency as an indicator of the semantic

closeness between terms. To address the building of an efficient text summarization system,authors in [61] proposed to use a lexical thesaurus, namely WordNet, to generalize concepts and, thus, to identify the topics within a text document, while, in [100], a taxonomy is constructed to provide a structure for linking similar concepts in different dissertation abstracts. Differently, in [49] a system for the construction of taxonomies that yields high accuracy with automated categorization systems on the Web is presented. Differently, the focus of the approach presented in [32] is to support the construction of schema ontologies and their corresponding instances, based on Description Logics, by exploiting both functional dependencies [70] and association rules [2]. A more detailed description of the proposed method is given in Chapter 7.

2.4 Data mining applications

This section overviews main data mining systems relative to three of the application contexts in which the use of generalized patterns has been already investigated: context-aware analysis (see Section 2.4.1), social network data analysis (see Section 2.4.3), and network traffic analysis (see Section 2.4.2).

2.4.1 Context-aware systems

Context-aware applications allow service providers to adapt their services to the actual user needs, thus offering them personalized services depending on their current application context. Provided services could be personalized by exploiting either the current context of the user [27, 66], or historic context and behavior of the user [30]. An effective user and service context profiling focuses on supporting user activities and service provisioning by means of a general-purpose system able to tailor service supply with high flexibility.

Research activities on context-awareness computing have been devoted both to providing different definitions of context-awareness and to building different context-aware applications (e.g., mobile phone applications [50], medical applications [113], computer vision applications [120]). Context consists of any circumstantial factors or application context users are involved in. Thus, context-awareness means that the system is able to use context information. A system is context-aware if it can extract, interpret and use context information and adapt its functionalities to the current usage context.

An in-depth literature review focused on context-aware systems has been presented and discussed in [60], in which a classification of the most significant related papers is proposed. In particular, the analyzed papers are classified according to the architectural layer of a general context-aware system to which each paper refers (e.g., application layer, middleware layer, network layer). Differently, in [25] the authors propose a survey focused on data-oriented context models. Firstly, a set of currently available context models is described, then a comprehensive evaluation framework is introduced to allow application designers to select the appropriate context model for a given target application. Main contextual information retrieval evaluation methodologies are also discussed in [106].

The usage of statistical and machine learning techniques (e.g., rule induction, neural networks, Bayesian networks) or data mining techniques (e.g., classification algorithms) has been proposed to exploit context data in building more accurate predictive user models (e.g., [89], [110], and [124]) and user profiles (e.g., [88]). Usually, different service and application models are tailored to the user and to the situation in which she/he is involved by means, for example, of rule based or probabilistic approaches. These models are then exploited to (i) suggest applications and services depending on what are the user interests in his/her current situation [112], (ii) perform customer segmentation [67], and (iii) personalize information retrieval tasks [41, 74]. Differently from previously cited works, in [18] the problem of user and service profiling is addressed by means of generalized association rules, which are classified in semantic groups by exploiting a set of rule templates. A more through description of the proposed system is reported in Chapter 4.

Integration of contextual information and data mining techniques is also discussed in [102]. However, in the proposed context-based data mining framework, the context information is exclusively used to integrate multiple datasets thus allowing consolidated mining on multiple physical datasets. Hence, the context knowledge neither directly drives the mining algorithms nor guides the selection of the mined patterns. Contextual information has been recently adopted in [102] for interactive postmining of association rules [2] driven by both ontologies and ad-hoc rule schemas [81] representing, respectively, user knowledge and expectations.

Context-awareness in the specific domain of mobile applications has been addressed from many different points of view. For example, context information has been used to analyze the collaboration between a mobile terminal user and another party [50] (i.e., another user or a mobile service), or to address context-based data reduction in mobile environments [56] for tailoring

a service to both the current user context and the characteristics of the used mobile device. The analysis of the logs containing user locations, provided by the mobile devices, has been also performed by means of data mining algorithms. For example, in [14] a location-based recommendation system for mobile devices is presented. It performs context-based data mining by means of a decision tree classification algorithm. Generalized association rule extraction has been successfully employed in context-aware domain. For instance, in [112] context-aware service and location pattern discovery in mobile Web environments are both addressed by means of multiple-level association rules. Similarly, in [18] a framework, namely CASMine, for context-aware user and service profiling is presented. Unlike [112], it performs a support-driven generalized association rule extraction, based on the algorithm first proposed in [16]. The application of some of the reasearch works presented in this thesis to real-life context-aware mobile systems is discussed in Chapters 4, 5, and 6.

2.4.2 Network traffic analysis

A relevant effort has been devoted to the application of data mining techniques in network traffic domain [68]. Related works commonly analyze network traffic captures of network streams for (i) network monitoring purposes, (ii) service profiling, (iii) bandwidth shaping, or (iv) intrusion detection. For instance, research activities include studying correlation among data [15, 96, 46] mining information for prediction [47, 85] or grouping network data with similar properties [45].

Frequent itemsets and association rules mining are shown to be promising techniques to highlight hidden knowledge in network flows [15]. NED (NEtwork Digest Framework) [39] is an efficient tool to characterize traffic data and detect anomalies by means of continuous queries, traffic filtering and refinement analysis (i.e., association rules). In [8] authors address traffic characterization by means of generalized association rule extraction. It exploits a generalized itemset mining similar to that first proposed in [16] and applies it network captures obtained from continuous queries. Some applications of the generalized itemsets in the context of network traffic monitoring and bandwidth shaping are reported in Chapters 4 and 5.

2.4.3 Social network analysis

Online communities provide a powerful resource suitable for discovering relevant social knowledge by means of data mining algorithms and tools tailored to most common user interests. In the last years, many different research efforts have been devoted to (i) developing new recommender systems to enhance the quality of product suggestions to the customers, (ii) improving the understanding of online resources by means of the categorization of UGC, and (iii) building query engines that take advantage of emerging semantics in social networks. For example, recommender systems are focused on identifying the objects (e.g., products, news) that highlight the highest correlation with the user behavior and preferences. Different approaches have been proposed to characterize the users and the items to be suggested. For example, in [119] a news recommendation system that also takes into account the opinion of readers is discussed. For each news discussion thread that has been posted on a social bookmarking site a topic profile is built. An extension of the previous approach exploited a graph-based representation to model the content similarity between comments and logic relationships among them [76]. Recommendation systems can be also exploited to enhance the quality of tags by analyzing the interest of a single user and/or the trend of social communities. For example, in [22] a classification system identifies the documents and the associated tags that are more likely to be of user interest. Differently, in [99] the authors exploited a hierarchical clustering algorithm to identify the correlation among groups of tags in order to reduce tag redundancy and ambiguity. To improve the quality of suggested tags for photo annotation, in [101] the authors proposed a method based on two steps. Given a photo with user-defined tags, for each tag an ordered list of candidate tags is first provided based on co-occurrence measure. Next, the lists of candidate tags are aggregated to generate a ranked list of recommended tags. An interesting overview of the most popular algorithms for computing the similarity among users of the online communities based on the recommended items is presented in [55].

The user-generated content has been extensively studied to improve the understanding of online resources (e.g., Web pages, photos, videos). For example, the discovery of most relevant social interests and the categorization of Web objects have been both addressed in [78] by exploiting tags assigned by users. In [121] proposed an optimization framework to assign the correct category to web resources is proposed. It focused on representing, by means of a graph-based model, the relationships between Web objects and social tags provided by del.icio.us and propagating the category information of

training samples from one domain to another.

Other kinds of applications are targeted to advanced analysis of UGC retrieved from online communities. As pointed out by [57], query engines can take advantage of folksonomies and ontologies extracted from social networks sites. For example, in [24] a framework to improve query results according to the user interest is proposed. The social relationships between users and the correlation among tags associated with media resources are represented by a unified graph model. Finally, the information provided by posts on social network sites has been exploited to improve the quality of online news searches. In [1] Twitter messages are exploited to retrieve keywords that are highly correlated with the user query. In particular, the authors analyzed the geographical content of Twitter message to determine which messages are more relevant according to the user query and retrieve a set of semantically related keywords. Differently, the system proposed in [34, 33] addresse Twitter post characterization through static and dynamic generalized association rule mining by considering both tweet keywords and related contextual features, including geographical information. A more detailed description of the proposed frameworks is reported in Chapters 4 and 6.

Chapter 3

Generalized association rule mining

Generalized association discovery focuses on discovering associations, i.e., patterns, among data at different abstraction levels by exploiting hierarchical semantic model, i.e., taxonomies, to aggregate knowledge into higher level concepts.

In this chapter the generalized itemset and association rule mining problems are formally stated.

The Chapter is organized as follows. Section 3.1 introduces preliminary definitions and notations, while Section 3.2 formally states the problems of frequent generalized itemset and association rule mining.

3.1 Preliminary definitions

In this section, the main notions concerning generalized itemset mining from structured data are introduced. Definition 1 formally describes the input structured data on the top of which the generalized itemsets and rules are mined.

Definition 1 Structured dataset. *Let $\mathcal{T}=\{t_1, t_2, \ldots, t_n\}$ be a set of data features and $\Omega=\{\Omega_1, \Omega_2, \ldots, \Omega_n\}$ the set of corresponding domains. t_i may be either a categorical or a numerical discrete feature*[1]*. A structured dataset*

[1] *In the case of structured datasets with continuous attributes, the value range should be discretized into intervals, and the intervals mapped into consecutive positive integers.*

D is a collection of records, where each record r is a set of pairs (t_i,$value_i$) where $t_i \in \mathcal{T}$ and $value_i \in \Omega_i$. Each $t_i \in \mathcal{T}$, also called attribute, may occur at most once in any record.

Table 3.1 shows the structured dataset exploited as a running example. It reports some pieces of information coming from a business-oriented scenario, in which different factories submit orders to their supplier. For each order, the *order date*, the name of the *factory* who submitted the order, and the *city* in which it is located are stored.

Order date	**Factory name**	**City**
2010-06-17	Factory_A	Turin
2009-09-01	Factory_A	Turin
2010-10-20	Factory_B	Cambridge
2010-05-01	Factory_C	Rome
2010-05-01	Factory_C	Tivoli
2010-10-20	Factory_B	Asti
2010-10-20	Factory_A	Cuneo
2010-05-01	Factory_A	Cuneo
2010-10-20	Factory_C	Alba

Table 3.1: Structured dataset - Running example.

To generalize concepts represented in a structured dataset at a higher level of abstraction, the notions of generalization hierarchy and taxonomy are introduced. To formally state, in Definition 2, the concept of generalization hierarchy, the definition of rooted labeled tree is recalled first. A rooted labeled tree is an acyclic connected graph in which a node is selected as the root. A rooted labeled tree T could be denoted as $T(r, N, Label, E)$, where (1) N is the set of nodes; (2) $r \in N$ is the root node; (3) $Label$ is the set of node labels, for any node $n \in N$, $Label(n)$ is the label of node n; and (4) $E = \{(x, y) \mid x, y \in N, x \neq y\}$ is the set of edges.

A generalization hierarchy (see Figure 3.1(a)) is a rooted labeled tree used to represent how the values of an attribute domain are aggregated into higher level values/concepts. Each leaf node belonging to the tree reported in Figure 3.1(a) represents a distinct value of the domain of the *order date* attribute of the running example, while each non-leaf node represents a generalization (higher level concept) of its children which may be further generalized by its parent.

Definition 2 Generalization hierarchy. *Let t_i be a data feature and Ω_i its domain. A generalization tree GT_i is a hierarchy of generalizations built over values in Ω_i and it is represented by a rooted labeled tree $GT_i(r, N, Label, E)$ where*

- *the set of labels Label is a superset of Ω_i (i.e., $\Omega_i \subseteq Label$) and includes both the values in the attribute domain and their generalizations,*
- *leaf nodes in GT_i are labeled with values in Ω_i,*
- *non-leaf nodes are aggregations of the values in Ω_i and are labeled with values not in Ω_i,*
- *the root node is labeled with the special value $\perp$ (i.e., undefined).*
- *for each label $l \in Label$ there is one and only one node in GT_i labeled with l.*

Figure 3.1 reports three examples of generalization hierarchies built over the attributes of the dataset D shown in Table 3.1. The *order date* attribute has the canonical form YYYY-MM-DD and might be generalized into its corresponding month, semester, and year (see Figure 3.1(a)), while the *city* attribute might be generalized into region/country and state (see Figure 3.1(b)). Since the *factory name* attribute has no meaningful aggregations, the relative GT_3 aggregates all leaves labeled with values in Ω_3 directly in the root node (see Figure 3.1(c)).

A taxonomy is defined as a set of generalization hierarchies on the attributes of a structured dataset. For instance, the set of the three generalization hierarchies reported in Figure 3.1, is a taxonomy defined on the attributes of the running example dataset.

Definition 3 Taxonomy. *Let $\mathcal{T}$ be a set of attributes. A taxonomy $\Gamma = \{GT_1, GT_2, \ldots, GT_n\}$ is a forest of generalization hierarchies, where GT_i is a generalization hierarchy defined on the attribute $t_i \in \mathcal{T}$.*

Albeit a taxonomy may include an arbitrary set of generalization hierarchies for each attribute, for the sake of simplicity in the following we let Γ contain one and only one generalization hierarchy for each attribute.

By means of Definitions 4-12 we formally define the concept of itemset and its main characteristics and properties. To this aim, we first introduce the concept of item as a couple (attribute,value). For example, (city, Turin) and (city, Italy) are two examples of items.

Definition 4 Item. *Let $\mathcal{T}$ be a set of attributes and Γ a taxonomy built on $\mathcal{T}$. An item is a pair (t_i, value$_i$) where $t_i \in \mathcal{T}$, value$_i$ is the label of one node of $GT_i \in \Gamma$, and value$_i \neq \perp$.*

An itemset is a set of items (e.g., {(date,2010-06-17), (city,Turin), (factory name,Factory_A)} and {(date,may 2010), (city,Lazio)}). In Definition 5 the concept of itemset in the context of structured data is defined.

Definition 5 Itemset. *Let $\mathcal{I}$ be the enumeration of all items. An itemset $X \subseteq \mathcal{I}$ is a set of items such that each attribute t_i may occur at most once in X.*

In the context of structured data, an itemset is called k-itemset if it contains exactly k items, each one belonging to a distinct data attribute.. For example, {(date,may 2010), (city,Lazio)} is a 2-itemset, while {(date,2010-06-17), (city,Turin), (factory name,Factory_A)} is a 3-itemset.

Generalized itemsets represent high level recurrences hidden in the analyzed data. A generalized itemset of length k (i.e., a generalized k-itemset) is a set of generalized or not generalized items including at least one generalized item. A more formal definition follows.

Definition 6 Generalized itemset. *Let $\mathcal{T}$=$\{t_1, t_2, \ldots, t_n\}$ be a set of attributes belonging to a structured dataset and $\mathcal{I}$ the enumeration of all the items in the corresponding dataset. Let Γ be a taxonomy over $\mathcal{T}$, and $\mathcal{E}$ the set of generalized items derived by all the generalization hierarchies in Γ. A generalized itemset Y is a subset of $\mathcal{I} \bigcup \mathcal{E}$ including at least one generalized item in $\mathcal{E}$. Each attribute $t_i \in \mathcal{T}$ may occur at most once in Y.*

In the following, four different (generalized) itemset characteristics and properties of interest are enumerated. In particular,

1. *itemset matching and support* (Cf. Definition 7), exploited to evaluate the frequency of occurrence of a given generalized itemset in the source data (Cf. Definitions 7 and 8),
2. *itemset level* (Cf. Definition 9), exploited to categorize itemsets depending on their generalization grade,
3. *well-formed itemset* (Cf. Definition 10), exploited to identify a subset of level-sharing itemsets during the mining process, and

4. *itemset ancestor, direct ancestor, and descendant* (Cf. Definition 11), exploited to define the relationships among itemsets at different abstraction levels.

To identify which dataset records are supported (covered) by a (generalized) itemset we introduce the concept of (generalized) itemset matching. An (generalized) itemset matches a given record if all its items either belong to the record or are ancestor nodes, in the corresponding generalization hierarchy, of a record item.

Definition 7 (generalized) Itemset matching. *Let X be a (generalized) k-itemset, D a structured dataset, and Γ a taxonomy. A (generalized) itemset X matches a record $r \in D$ if and only if $\forall$ (t_i, value$_i$) $\in X$:*

1. *(t_i, value$_i$) $\in r$, or*

2. *value$_i$ is the label of a non-leaf node of the generalization hierarchy GT_i defined on t_i, such that it exists a path from that node to a leaf node h and (t_i, Label(h)) $\in r$.*

Definition 8 (Generalized) itemset support. *Let D be a structured dataset and Γ a taxonomy built over D. The support of a generalized itemset X is given by the number of records $r \in D$ matching X divided by the cardinality of D.*

For example, the itemset {(date,2010-06-17), (city, Turin)} has support equal to $\frac{1}{5}$ as it matches the first record in the running example dataset (in Table 3.1), while the generalized itemset {(date,may 2010),(city, Lazio)} has support $\frac{2}{5}$ as it matches the fourth and the fifth records.

Generalized itemsets may be classified based on the abstraction level of their items. The level L[(t_i, value$_i$)] of an item (t_i, value$_i$) is given by the height of the node labeled with value$_i$ in GT_i plus 1, i.e., it is the length of the longest downward path to a leaf from that node plus 1. Consider, for instance, the generalization hierarchy GT_2 on the city shown in Figure 3.1(b). The level of the item (city, Italy) is 3 as the longest path on GT_2 from Italy to a leaf node has length 2, while the level of (city, Turin) is 1. Definition 9 extends the notion of item level to (generalized) itemsets. The level of an (generalized) itemset X is defined as the maximum level of the items in X.

Definition 9 (generalized) itemset level. *Let X be a k-itemset. The level $L[X]$ of a (generalized) itemset is the maximum item level by considering items in X, i.e., $L[X] = max_{1 \leq j \leq k}\{L[(t_j, value_j)]\}$.*

For instance, the level of {(date,2010-06-17), (city,Italy)} is 3.

Authors in [52] first propose to exploit the generalized itemset level to mine, from transactional data, only the subset of itemsets composed of item of the same level, i.e., the level-sharing generalized itemset. For example, {(date,2010-06-17), (city,Turin)} is a level-sharing generalized itemset, while {(date,2010-06-17), (city,Italy)} is not. However, the notion of level-sharing generalized itemset is proposed for transactional data and is based on one single generalization hierarchy defined on the whole set of possible items. When considering structured datasets and a taxonomies composed of many generalization hierarchies characterized by different heights (consider, for instance, the three GT_i reported in Figure 3.1), A similar notion of well-formed itemset, which is customized on structured data and allows dealing with taxonomies composed of unbalanced generalization hierarchies (i.e., taxonomies characterized by different heights like the one reported in Figure 3.1) is introduced.

Definition 10 Well-formed Itemset. *Let X be a k-itemset, m=$L[X]$ the level of X, and Γ a taxonomy. X is a well-formed itemset if and only if $\forall\ x_i \in X$:*

1. *$L[x_i] = m$, or*
2. *$L[x_i] < m$ and x_i is a child of the root node for GT_i.*

Condition (A) in Definition 10 is satisfied by itemsets composed of items of the same level, while Condition (B) is satisfied by itemsets composed of items belonging to different levels and related to unbalanced generalization hierarchies. For instance, {(date,may 2010),(city,Piemonte)} is a well-formed itemset, according to condition (A), as the level of its items is 2. Differently, itemset {(date,year 2010), (city,Italy)} does not satisfy condition (A) as L[(date,year 2010)] = 4 while L[(city,Italy)] = 3. However, according to Definition 10 - condition (B), it is well-formed as well. In fact, the node labeled Italy is a child of the root node, while there is no level-4 items in the generalization hierarchy defined on the city attribute (see Figure 3.1). Thus, it does not exist any level-4 generalization of (city,Italy).

Dependencies among similar itemsets of different levels are expressed by the ancestor/descendant relationships. For example, the generalized itemset {(date,may 2010), (city,Piemonte)} is an ancestor of both {(date,2010-05-01), (city,Turin)} and {(date,2010-05-01), (city,Piemonte)}.

Definition 11 (Generalized) itemset ancestor and descendant. *Let $X, Y \subseteq \mathcal{I}$ be two (generalized) k-itemsets and Γ a taxonomy. X is an ancestor of Y on Γ, denoted as $X \in Anc[Y, \Gamma]$, if and only if $\forall$ item $y_i \in Y$ exists an item $x_i \in X$ such that either x_i is an ancestor of y_i in GT_i or $x_i = y_i$. If X is an ancestor of Y, then Y is a descendant of X, denoted as $Y \in Desc[X, \Gamma]$.*

Since the generalization process over an itemset X is typically performed by climbing up the corresponding taxonomy stepwise to obtain the first well-formed ancestor of X we introduce the notion of direct ancestor as the first well-formed ancestor itemset. For instance, both Y={(date,may 2010), (city,Piemonte)} and Z={(date,1st Semester 2010), (city,Italy)} are ancestors of X={(date,2010-05-01), (city,Turin)}. However, Y={(date,may 2010), (city,Piemonte)} is the direct ancestor of X={(date,2010-05-01), (city,Turin)} because Y is the first well-formed ancestor of X generated by climbing up the taxonomy stepwise over the items of X.

Definition 12 (generalized) itemset direct well-formed ancestor. *Let $X, Y \subseteq \mathcal{I}$ be two k-itemsets and Γ a taxonomy. X is a direct ancestor of Y over Γ if and only if*

1. *$X \in Anc[Y, \Gamma]$, and*
2. *X is a well-formed itemset, and*
3. *$\forall\ Z \in Anc[Y, \Gamma]$, if Z is well-formed and $Z \neq X$ then $L[X] < L[Z]$.*

Two (generalized) itemsets are disjoint if their corresponding item attribute sets are disjoint.

Definition 13 Disjoint (generalized) itemsets. *Let A and B be two arbitrary generalized itemsets. A and B are disjoint iff attr(A) $\cap$ attr(B) $= \emptyset$.*

Association rules are implications in the form $A \Rightarrow B$ where A and B are disjoint generalized itemsets.

Definition 14 Generalized Association Rule. *Let A and B be two disjoint (generalized) itemsets. A generalized association rule is represented in the form $A \Rightarrow B$, where A and B are the body and the head of the rule respectively.*

A and B are respectively denoted as antecedent and consequent of the generalized rule $A \Rightarrow B$. Generalized association rule discovery is commonly driven by the rule support quality index, whose formal definition follows.

Definition 15 Generalized Association Rule Support. *Let $A \Rightarrow B$ be a generalized association rule. Its support s is the support of the generalized itemset $A \cup B$.*

In general, the support represents the prior probability of A and B (i.e., its observed frequency) in the source dataset.

Definition 16 Generalized Association Rule Confidence. *Let $A \Rightarrow B$ be a generalized association rule. Its confidence c is given by $\frac{s(A \cup B)}{s(A)}$.*

The confidence of a rule $A \Rightarrow B$ is the conditional probability of the generalized itemset B given the generalized itemset A.

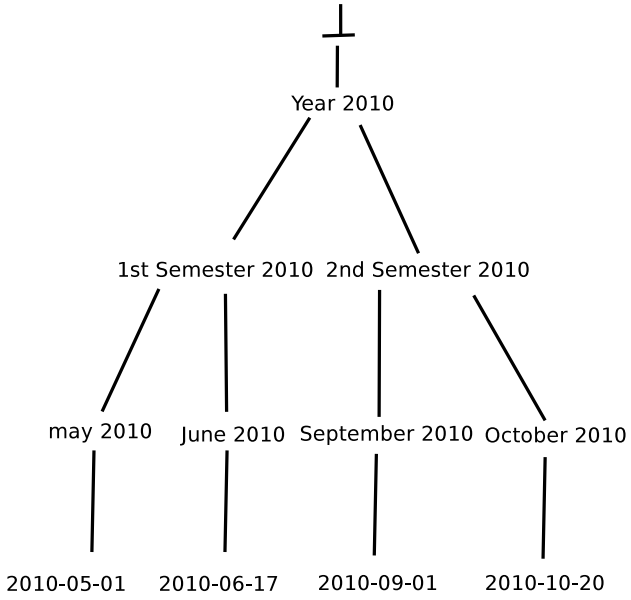
For example, the following generalized association rule states the service FlightStat, offered by a Web provider, is frequently asked by its visitors during the time slot [1 p.m., 4 p.m.].

(`service`: FlightStat $\rightarrow$ `time`: from 1 p.m. to 4 p.m.) ($s = 10\%$,$c = 88\%$)

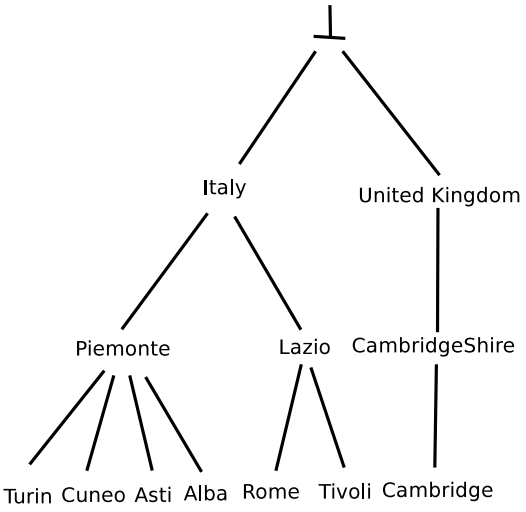
3.2 Problem statement

Given a structured dataset D, a taxonomy Γ, a minimum support threshold min_sup, and a minimum confidence threshold min_conf the generalized association rule mining problem entails the extraction of all generalized and not generalized association rules (Cf. Definition14) that satisfy both min_sup and min_conf.

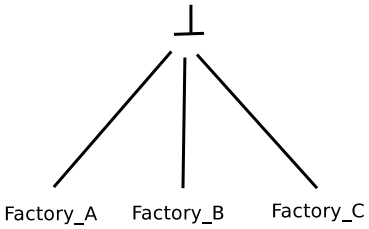
Generalized association rules are commonly discovered by means of a two-step process: (i) Frequent generalized itemset extraction and (ii) rule generation from the extracted frequent itemsets. The first step entails the extraction of all (generalized) itemsets (Cf. Definition6) that satisfy min_sup.



(a) Date - Generalized Tree GH_1



(b) City - Generalized Tree GH_2



(c) Factory name - Generalized Tree GH_3

Figure 3.1: Generalization hierarchies for the attributes in the example dataset

Chapter 4

The GENeralized Itemset DiscOverer algorithm

This chapter presents a generalized itemset mining algorithm, namely GENIO (GENeralized Itemset DiscOverer) that performs an opportunistic knowledge aggregation, driven by the given taxonomy, to prevent the extraction of redundant high level patterns. The chapter is organized as follows. Section 4.1 thoroughly describes the GENIO algorithm. Section 4.2 evaluates the performance of the GENIO algorithm on data coming from the network traffic domain. Sections 4.3 and 4.4 present two data mining systems, based on the GENIO algorithm, to perform knowledge discovery from context-aware and social network data. Furthermore, they also report an experimental assessment of the effectiveness of the proposed systems in the corresponding application contexts. Finally, Section 4.5 evaluates the scalability of the proposed algorithm on synthetic data in terms of execution time.

4.1 The GENIO Algorithm

The GENIO (GENeralized Itemset DiscOverer) algorithm extracts frequent generalized itemsets. However, it mines a generalized itemset if and only if at least one of its descendants is infrequent with respect to the minimum support threshold. It exploits a taxonomy Γ (i.e., a set of generalization hierarchies of arbitrary heights) to generalize concepts defined in the structured dataset under analysis.

GENIO takes in input a structured dataset D, a taxonomy Γ built over D, and a minimum support threshold min_sup. It discovers all frequent item-

Algorithm 1 Generalized Itemset Discoverer

Input: minimum support *min_sup*, taxonomy Γ, dataset D
Output: L, set of generalized frequent itemsets
1: $k = 1$, $L = \emptyset$
2: C_1 = set of items in D
3: **repeat**
4: scan D and count support for each $c \in C_k$
5: $Gen = \emptyset$ // generalized itemset container
6: **for all** c in C_k **do**
7: **if** support of $c <$*min_sup* **then**
8: *new_gen_itemset* = taxonomy_evaluation(Γ, c)
9: update *Gen* with *new_gen_itemset*
10: **end if**
11: **end for**
12: **if** $Gen \neq \emptyset$ **then**
13: scan D and count support for each itemset in *Gen*
14: **end if**
15: L_k = { itemsets in $\{C_k \cup Gen\}$ whose support $\geq$ *min_sup* }
16: $k = k + 1$
17: C_k = candidate_generation(L_{k-1})
18: **until** $C_k \neq \emptyset$
19: **return** L

sets and generalized itemsets having at least an infrequent descendant. The generalization process, driven by the taxonomy Γ, is support-driven, i.e., it generalizes an itemset only if it is infrequent with respect to the minimum support threshold. The pseudo-code of GENIO is given by Algorithm 1. GENIO implementation is based on *Apriori* [2].The *Apriori* algorithm is a level-wise algorithm that, at each iteration, generates all frequent itemsets of a given length. At arbitrary iteration k, two steps are performed: (i) Ccndidate generation, the most computationally and memory intensive step, in which all possible k-itemsets are generated from *(k-1)*-itemsets, (ii) candidate pruning, which is based on the property that all subsets of frequent itemsets must also be frequent, to discard candidate itemsets which cannot be frequent. Finally, actual candidate support value is counted by scanning the dataset.

GENIO follows the same level-wise pattern generation. However, it entails (i) aggregating knowledge associated with rare itemsets into higher level concepts by lazily evaluating the taxonomy Γ (lines 6-11), and (ii) exploiting the characteristics of the structured datasets to prune candidates effectively (line 17). Once the support value of each candidate itemset in C_k has been computed (line 4), the generalized versions of the infrequent ones are generated by evaluating the taxonomy (line 8) and included in the *Gen* set (line 9). In particular, by applying on each item (t_j, $value_j$) in itemset c the corresponding generalization hierarchy RR_j, the generalized versions of each item in c are generated (line 8). All the itemsets obtained by replacing one or more items in c with their generalized versions are generalized itemsets of c.

Hence, the taxonomy evaluation process applied on the itemset c may potentially generate many generalized itemsets. The generalization process of c is triggered if and only if c is infrequent with respect to the minimum support threshold (line 7). Once triggered, the generalization process is repeated until a complete taxonomy evaluation has been performed. For example, let {(*destination-IP-address*, 130.192.15.17), (*destination-port* = 21)} be an infrequent itemset in C_k. By applying on the destination port the generalization hierarchy RR_{port} shown in Figure 5.9(a) and on the destination IP address $RR_{IP-address}$ shown in Figure 4.1, the following generalized itemsets are generated:

- {(*destination-IP-address* = 130.192.15.0/24), (*destination-port* = 21)}
- {(*destination-IP-address* = 130.192.15.17), (*destination-port* = Well-known)}
- {(*destination-IP-address* = 130.192.15.0/24), (*destination-port* = Well-known)}

The insertion of redundant generalized itemsets (i.e., generalized itemsets previously generated by different infrequent descendants) in *Gen* is prevented by the update procedure in line 9. If the *Gen* set is not empty, the support value of each generalized itemset in *Gen* is computed by performing a further scan of the source dataset (line 13).

During the candidate generation step (line 17), GENIO exploits the uniqueness of attributes in a given record of a structured dataset to further prune candidate itemsets (e.g., in the traffic network domain a flow, i.e., a record of the network trace, with multiple *destination-IP-address* cannot be defined). For example, suppose that after the first dataset traversal, we have m frequent 1-itemsets tagged *source-IP-address* and n tagged *destination-IP-address*. Apriori exhaustive candidate generation would produce $\binom{m+n}{2}$ possible combinations. Since each attribute could appear only once in each record, only $m \cdot n$ 2-itemsets (obtained by combining one item tagged *source-IP-address* and one item tagged *destination-IP-address*) are relevant combinations. Hence, GENIO generates only this subset of candidates. Thus, the required computational and memory cost is reduced.

ID	Number of records	Number of different items
D1	18051	32143
D2	17374	30617
D3	16783	30072
D4	3802	9350
D5	2074	5825

Table 4.1: Characteristics of the network traffic datasets

4.2 Experimental evaluation of GENIO on network traffic data

The performance of the GENIO algorithm has been evaluated in the context of network traffic analysis by means of a large set of experiments. In particular, they analyzed (i) the performance of the frequent generalized itemset miner and (ii) the number and the relevance of the extracted itemsets.

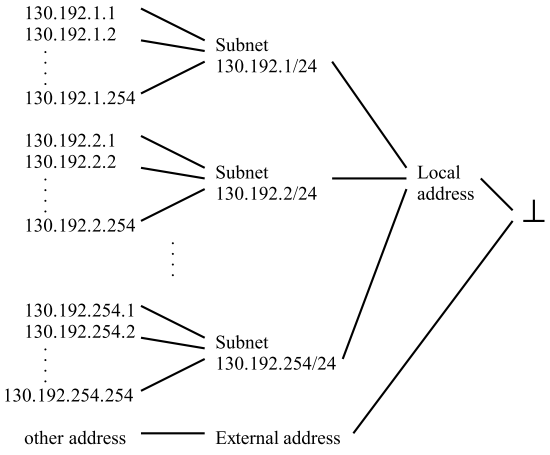


Figure 4.1: A generalization hierarchy $RR_{IP-address}$ for the source and destination IP address data attributes

4.2.1 Experimental setting

Experiments have been performed on an AMD Sempron™ 2400+ PC with 1666 MHz CPU and 512 Mb main memory, Linux operating system. Five real datasets have been exploited to validate the efficiency and effectiveness

of the GenIO algorithm. These datasets were obtained by performing different capture sessions using the open-source Network Analyzer tool [87] on a backbone link of the campus network. Captured traffic has been aggregated in traffic flows, i.e., records which summarize a group of similar and temporally contiguous packets. Each flow is a data record characterized by six attributes: Source IP address, destination IP address, source port, destination port, flow size (i.e., the size of the flow expressed in byte), and number of IP packets aggregated in that flow. We will refer to each dataset by using the ID shown in the first column of Table 4.1. Table 4.1 reports the number of records and the number of different items for each dataset. The used datasets have significantly different characteristics in terms of the cardinality and the number of different items. Dataset ids are sorted by decreasing cardinality.

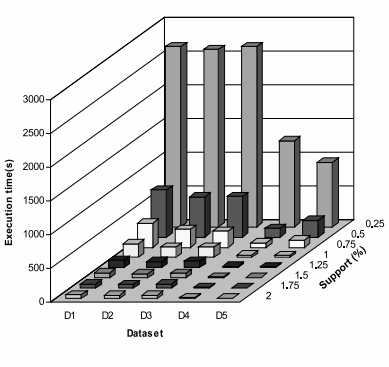
The taxonomy used in the experiments aggregates infrequent items according to the following generalization hierarchies: (1) source and destination ports are aggregated by exploiting the generalization hierarchy shown in Figure 5.9(a), which introduces three established aggregation values (i.e., *well-known, registered, dynamic*). (2) Source and destination IP addresses are aggregated by exploiting the generalization hierarchy shown in Figure 4.1. IP addresses are aggregated in *subnet* if they are local to the campus network. IP addresses which do not belong to the campus network are aggregated in a more general *external address* node. Furthermore, both the flow size (bytes) and number of IP packets attributes are uniformly discretized in 4 bins, whose intervals are [1,1,000), [1,000, 2,000), [2,000, 3,000), and equal or greater than 3,000.

4.2.2 Frequent generalized itemset extraction performance

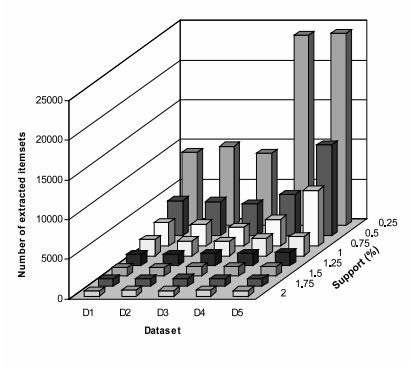
To evaluate the performance of the GenIO algorithm, the following issues have been addressed: (i) the performance of the proposed generalized itemset miner, in terms of execution time, number of extracted itemsets, and aggregation impact factor, (ii) comparison between GenIO and an optimized version of the well-known multiple-level algorithm, namely Cumulate, proposed in [103], in terms of time reduction and pruning selectivity.

The GenIO algorithm performance

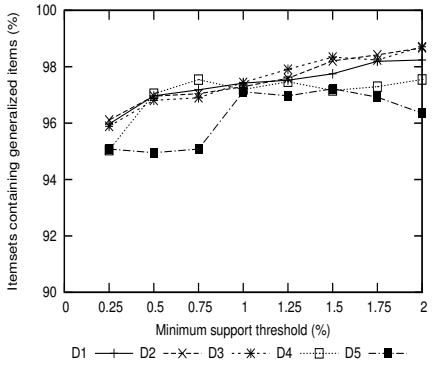
Figure 4.2 shows the execution time (Figure 4.2(a)), the number of extracted generalized itemsets (see Figure 4.2(b)), and the aggregation impact factor



(a) Execution time



(b) Number of extracted itemsets



(c) Aggregation impact factor

Figure 4.2: Performance of the GENIO algorithm

(see Figure 4.2(c)) yielded by the GENIO algorithm by enforcing different support thresholds. The execution time spent by GENIO in itemset mining, shown in Figure 4.2(a), is mainly due to the generalization process and the dataset scans. While the former factor depends on taxonomy features (e.g., number of aggregation levels), the latter is proportional to the number of transactions. Since the taxonomy exploited in the network traffic analysis is characterized by a few aggregation levels, the time spent for dataset scans is dominant. Furthermore, the execution time increases when larger datasets and lower support thresholds are enforced as the dataset scans require much more time.

Figure 4.2(b) shows the number of extracted generalized itemsets by en-

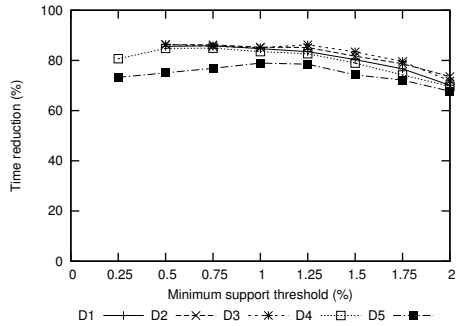


Figure 4.3: GENIO w.r.t. Cumulate: Time reduction (%)

forcing different support thresholds. When support threshold decreases the number of mined itemsets increases. For high support thresholds, the number of extracted generalized itemsets is quite constant. When low support thresholds are enforced, the number of generalized itemsets significantly increases especially when the data distribution becomes sparser (see datasets D4 and D5 in Figure 4.2(b)). This effect is given by the high number of low frequency items characterizing a sparse dataset. These items become frequent when the mining process is performed by enforcing a low support threshold.

The impact of the aggregation process on the number of mined itemsets has been also analyzed for each dataset. In particular, Figure 4.2(c) reports the percentage of itemsets achieved by the generalization process with respect to the total number of extracted itemsets by varying the minimum support threshold. Since the generalization process is a support-driven process, the number of itemsets containing terms at higher level of the taxonomy increases by enforcing higher support thresholds.

Comparison between GENIO and Cumulate

A comparison between GENIO and our optimized implementation of Cumulate [103], a traditional well-known generalized itemset mining algorithm, has been performed. Figure 4.3 shows the time reduction for generalized itemset extraction yielded by GENIO with respect to Cumulate. Experiments, on all considered datasets, have been performed by enforcing different support thresholds. Cumulate did not correctly terminate the extraction task on datasets D1, D2, D3 for the lowest considered support (see Figure 4.3). GENIO algorithm yields a significant reduction of the execution time for any considered datasets and for all support thresholds. The reduction is mainly

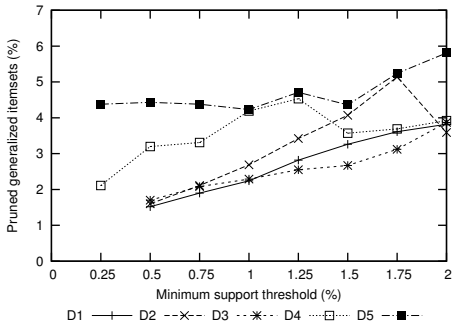


Figure 4.4: GENIO w.r.t. Cumulate: Pruned generalized itemsets (%)

due to the support driven opportunistic aggregation, which reduces the number of extracted generalized itemsets (as shown in Figure 4.4), and to the exploited strategy to further prune candidate itemsets (see Section 4.1).

To evaluate the pruning selectivity of our approach, the corresponding percentage of pruned generalized itemsets with respect to Cumulate is reported. Since the GENIO algorithm performs a support driven opportunistic aggregation of itemsets, it extracts a smaller number of generalized itemsets than Cumulate. Figure 4.4 shows the percentage of pruned generalized itemsets for all datasets. GENIO yields a good percentage of pruned itemsets for any considered support threshold. For high support thresholds, since the absolute number of generalized itemsets is relatively small (see Figure 4.2(b)) the corresponding percentage of pruned generalized itemset is about 4%-6% (see Figure 4.4). For low support thresholds, a high number of low frequency items become frequent. Thus, the absolute number of generalized itemsets significantly increases (see Figure 4.2(b)), while the percentage of pruned generalized itemsets (see Figure 4.4) preserves its trend. Hence, the GENIO algorithm significantly reduces the cardinality of the extracted knowledge. The Cumulate algorithm could obtain the same knowledge mined by GENIO only through a very expensive post-processing analysis over a very large set of itemsets.

4.2.3 Analysis of the extracted generalized itemsets

In the following the meaning and the usefulness of the extracted generalized itemsets in the network traffic domain are discussed.

Figure 4.5 shows the support of the generalized 2-itemsets in the form {*destination-IP-address, destination-port*} obtained from dataset D1. For

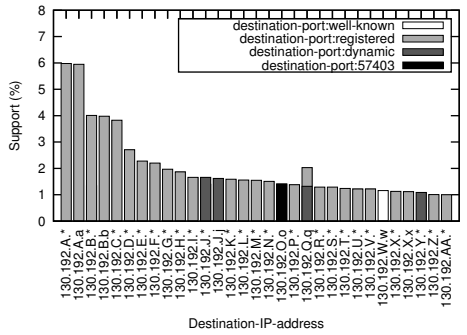


Figure 4.5: GenIO: support values of the extracted 2-itemsets for different destination IP addresses and ports

better visualization, results have been restricted to addresses of the campus network. Thus, no external IP address has been considered. The address is automatically aggregated into the corresponding subnet when the single IP address support value is below the minimum support threshold (set to 1%). Figure 4.5 provides a characterization of the traffic on the campus network. Many of the extracted itemsets describe general network features. For example, the most supported generalized itemset identifies the VPN concentrator of the campus network.

Larger generalized itemsets (i.e., itemsets with more than 2 items) can be exploited to focus the analysis on specific traffic behaviors. For example, the 4-itemsets {*(destination-IP-address, 130.192.O.o), (destination-port, 57403), (source-IP-address, x.x.x.x), (source-port, registered-port)*}, having support equal to 2.3%, highlights an unconventional high-volume traffic towards a specific host of the campus network, whereas the 4-itemsets {*(destination-IP-address, 130.192.A.a)*, (*destination-port*, *registered-port*), (*source-IP-address, y.y.y.y*), (*source-port, well-known*)}, having support equal to 2%, identifies connections to the VPN concentrator by means of a client using well-known source ports.

4.3 A context-aware application: The CAS-MINE Framework

The CAS-Mine framework is a context-aware environment, based on the GenIO algorithm, to perform both user and service profiling effectively. It allows shaping service provisioning by considering the current context of the

user. By discovering recurrent patterns on user habits, service providers can partition users into predefined categories over which service provisioning may be modeled and personalized. Furthermore, service profiling may allow providers to effectively shape service supply, promotions, and system size depending on the actual application usage.

CAS-MINE exploits the GENIO algorithm [16] to discover generalized association rules. The main architectural blocks of the CAS-MINE framework (shown in Figure 4.6) are described in the following.

Data collection and preprocessing. Context knowledge is provided by different and heterogeneous sources (e.g., mobile devices) collecting information about the context of the user submitting the request (e.g., GPS coordinates, temporal information), and the requested services (e.g., service description). Next, data is cleaned by removing irrelevant and redundant information and integrated into a common data structure.

Mining activity. The aim of the mining activity block is to discover interesting correlations and recurrent patterns in context data. In the CAS-MINE framework, interesting patterns are extracted in the form of *generalized association rules*, i.e., rules that represent general correlations among context data. The generalization step is performed by means of an analyst-provided taxonomy (i.e., is-a hierarchy) defined on structured data. Analysts should provide meaningful sets of generalization hierarchies based on their knowledge on context information concerning user requests collected in the source dataset. Context data can be aggregated at different granularity levels to discover more informative and compact knowledge in a flexible way.

The extraction of generalized association rules is performed by means of a (traditional) two-step process: (i) frequent generalized itemset extraction and (ii) rule generation from the extracted frequent itemsets. Since the GENIO algorithm [16] is known to be more efficient than previous approaches (e.g., [52]) in automatically extracting interesting generalized itemsets from structured data, CAS-MINE exploits GENIO to perform the first step. The extraction of generalized rules (i.e., the second step) is performed by our implementation of the rule mining step of the Apriori algorithm [2].

In-depth analysis. The aim of the in-depth analysis block is two-fold: (i) selection of the most relevant rules and (ii) rule classification. Currently, extracted patterns are ranked by exploiting the *lift correlation*

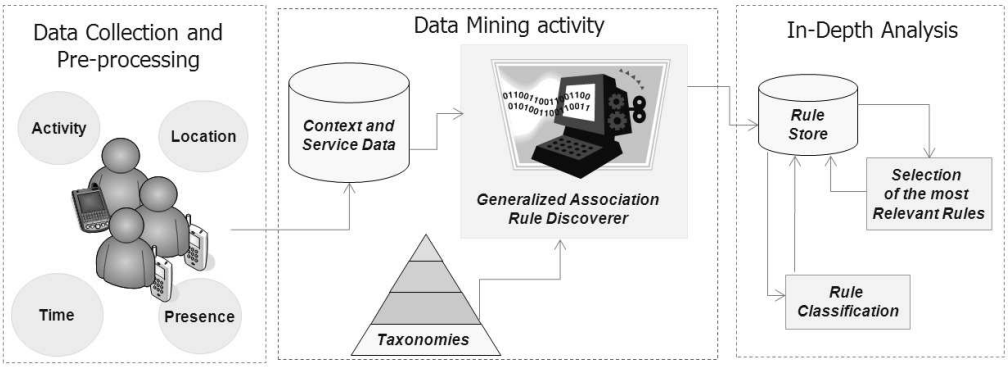


Figure 4.6: The CAS-MINE framework architecture

measure to effectively discriminate between reliable and unreliable patterns. The lift measure highlights rule correlation, thus overcoming the support and confidence well-known drawbacks without requiring complex computations. However, other data quality indices [107] could be easily integrated into the CAS-MINE framework as well.

The rule classification step categorizes the extracted rules according to their semantic interpretation in context-aware applications. Thus, extracted rules are partitioned in two main classes: user rules and service rules. User rules characterize user habits at different aggregation levels and allow service providers to offer personalized services tailored to the current user context. Service rules, instead, describe service characteristics and allow service providers to adapt service provisioning to the current context, independently of the requesting user. For each class, some relevant rule templates have been identified and discussed.

A more detailed description of each block and its functionalities is presented in the following sections.

4.3.1 Data Collection and Preprocessing

Since service requests typically depend on the requesting user environment, a collection of information describing the user context at request submission may enhance the quality of the provided services. The data collection and preprocessing block of the CAS-MINE framework handles data collection through services on mobile devices that collect user and service context information (e.g., temporal information, GPS coordinates, service description).

To ensure user privacy, context data retrieval and processing in a context-aware environment requires both the user informed consent on personal data treatment and compliance with laws in force.

Due to the distributed nature of mobile systems, separate logs have been recorded in different systems, describing different parts of the user activity. The data collection phase takes in input the raw context data provided by different, maybe heterogeneous, sources and join them into a unique data repository.

The preprocessing phase aims at making the raw input data fully compatible with the repository format. During the joining process, data are tailored to a common data structure by means of a data cleaning process. Data cleaning also discards useless or redundant information and correctly manages missing values. After preprocessing, the collected context information can be modeled as a structured dataset, where records represent service requests.

A structured context dataset is a structured dataset holding information on both service requests performed by different users, and the corresponding application context in which requests are submitted. Each record is a set of items describing a specific user service request. Attributes describe the represented information (e.g., *user identifier*, *service*, *time*) and take values (e.g., *ID54*, *weather*, *4:06 p.m.*) in the corresponding attribute domains.

4.3.2 Generalized association rule extraction

Generalized association rules provide a high level domain knowledge abstraction that allows a compact representation of general correlations among context data. In Chapter 3 the problem of generalized association rule mining is formally defined.

The discovery of the generalized association rules is driven by an analyst-provided taxonomy, i.e., a set of generalization hierarchies. Consider, for example, the *time* attribute, which defines the submission time of a service request. A simple generalization hierarchy that may be devised by a domain analyst is shown in Figure 4.7. The generalization hierarchy aggregates the submission time of the service request by 4-hour timeslots, and A.M./P.M. time periods. The root (represented as {}) aggregates all values allowed for the *time* attribute.

In the context-aware service profiling domain, many different generalization hierarchies may be defined for the *time, date, service, phone number,*

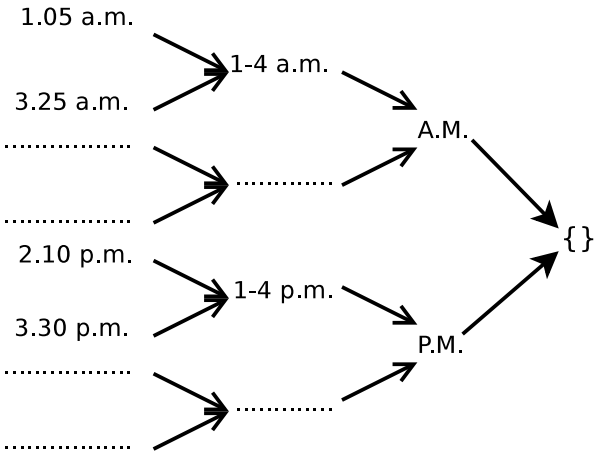


Figure 4.7: A generalization hierarchy RR_{time} for the time attribute

and *location* attributes. Analysts should provide meaningful generalization hierarchies based on their knowledge on context data collected into the source dataset.

Generalized association rules are discovered by means of a two-step process: (i) frequent generalized itemset extraction and (ii) rule generation from the extracted frequent itemsets. The first step exploits the GenIO algorithm [16] while the second step is performed by our implementation of the rule mining step of the Apriori algorithm [3]. The mining process is driven by the support threshold, which drives the itemset extraction process.

4.3.3 Supporting in-depth rule analysis

The in-depth analysis block of CAS-Mine addresses (i) the ranking of the most relevant rules by exploiting the lift quality index [107] and (ii) the classification of interesting rules useful for effectively supporting user and service profiling in context-aware applications.

Ranking relevant rules

Many quality measures [107] may support selection and ranking of the most interesting rules. The rules mined by CAS-Mine are sorted by means of the lift index [107], which measures the (symmetric) correlation between body

Table 4.2: CAS-MINE: length-2 user rule classes

Class	Question	Rule template	Example
AU-T	Given a user (or a user category), in what time period does he request services (or classes of services)?	$\{user\} \Rightarrow \{time\}$	$\{user = John\} \Rightarrow \{time = 6 - 7\ p.m.\}$ means that user John submits service requests between 6 and 7 p.m.
AU-D	Given a user (or a user category), in what period of the year does he request services (or classes of services)?	$\{user\} \Rightarrow \{date\}$	$\{user = John\} \Rightarrow \{date = winter\}$ means that user John submits service requests during the winter.
AU-P	Given a user (or a user category), where does he request services?	$\{user\} \Rightarrow \{place\}$	$\{user = John\} \Rightarrow \{place = OFFICE\}$ means that user John requests services in his office.
RS	Given a user or a user category, which services (or class of services) is he interested in?	$\{user\} \Rightarrow \{service\}$	$\{user = John\} \Rightarrow \{Service = SMS\}$ means that user John requests the SMS service.

and head of the extracted rules. The lift of a (generalized) association rule $A \Rightarrow B$ is defined as [107]

$$\text{lift(A, B)} = \frac{\text{c}(A \Rightarrow B)}{\text{s(B)}} = \frac{\text{s}(A \Rightarrow B)}{\text{s(A)s(B)}} \tag{4.1}$$

where $\text{s}(A \Rightarrow B)$ and $\text{c}(A \Rightarrow B)$ are, respectively, the rule support and confidence, and $\text{s}(A)$ and $\text{s}(B)$ are the supports of the rule antecedent and consequent. If lift(A,B)=1, the itemsets A and B are not correlated, i.e., they are statistically independent. Lift values below 1 show negative correlation, while values above 1 indicate a positive correlation between itemsets A and B.

Both positively and negatively correlated rules are selected by CAS-MINE to highlight interesting situations. For instance, positively correlated rules highlight the preferred user services, while negatively correlated rules identify the services used less than expected. Similarly, the interest of rules having a lift value close to 1 may be marginal. Hence, CAS-MINE ranks the mined rules according to their lift value to focus the analysis on the set of most (positively or negatively) correlated rules.

Rule Classification

The rule classification block categorizes correlated rules in classes to effectively address context-aware profiling. Since service providers are mainly interested in profiling both users and services, the CAS-MINE framework identifies two main classes of generalized association rules: (i) user rules and (ii) service rules.

To classify the extracted rules, we propose rule templates that define the general structure of interesting subsets of generalized association rules. All the rules that share the same template are characterized by the same attribute(s), but not necessarily the same values, in the body and in the head of the rule.

Definition 17 Generalized association rule template. *Let* $\mathcal{T}=\{t_1, t_2, \ldots, t_n\}$ *be a set of attributes. A generalized association rule template is represented in the form* $X \Rightarrow Y$*, where* X *and* Y *(the body and the head of the rule template, respectively) are two disjoint sets of attributes (i.e.,* $X \subseteq \mathcal{T} \wedge Y \subseteq \mathcal{T} \wedge X \cap Y = \emptyset$*).*

User rules User rules characterize user habits at any aggregation level. These rules allow service providers to offer personalized services depending on the current context of the user. Hence, the provided services can be adapted to actual user needs. In user rules the attribute "user" always appears either in the body or in the head of the rule.

User rules are further partitioned in more specific categories. For each category, a set of interesting rule templates is defined. For the sake of simplicity, rule templates are currently defined on length-2 and length-3 rules. The most interesting user rule templates are summarized in Table 4.2 and Table 4.3. They always include the user attribute in the body of the rule. For each template, an example and a short explanation are provided. User rules compliant with a given template describe user context knowledge at the appropriate[1] hierarchical level. Thus, they are suitable for context-aware user profiling.

Table 4.2 shows length-2 user rule classes. These classes are characterized by a single attribute both in the body and in the head of the rule. They have been further semantically partitioned into (a) common requested services (denoted as RS), defining the service type a user is mainly interested in, and (b) context of the requested service, identifying when or where users request for services (denoted with prefix AU). For the AU category three different rule templates have been defined. The AU-T template identifies the time period at which the user requests services. The AU-D template models the period of the year in which the user requests services. The AU-P template defines the location (place) of the user requesting services.

User rules not falling into the above classes may belong to two categories: (a) The rule is a specialization of rules belonging to templates in Table 4.2

[1]The appropriate level depends on the selected support threshold.

Table 4.3: CAS-MINE: length-3 user rule classes

Class	Question	Rule template	Example
AU-PT	Given a user (or a user category), where does he request services and in which time period?	$\{user\} \Rightarrow \{place, time\}$	$\{user, John\} \Rightarrow \{(place, office), (time, morning)\}$ means that user John requests application services during the morning in his office.
AU-PD	Given a user (or a user category), where does he request services and in which period of the year?	$\{user\} \Rightarrow \{place, date\}$	$\{(user, John)\} \Rightarrow \{(place, office), (date, winter)\}$ means that user John requests application services during the winter in his office.
AU-PPa	Given a user (or a user category), where does he request services and which service parameters are specified?	$\{user\} \Rightarrow \{place, param\}$	$\{(user, John)\} \Rightarrow \{(place, office), (param, OUT)\}$ means that user John requests outgoing application services in his office.
AU-DT	Given a user (or a user category), in what time and year periods does he request services?	$\{user\} \Rightarrow \{date, time\}$	$\{user = John\} \Rightarrow \{(date, June), (time, morning)\}$ means that user John requests application services during the morning in June.
AU-PaT	Given a user (or a user category), in what daily time periods does he request and which service parameters are specified?	$\{user\} \Rightarrow \{param, time\}$	$\{(user, John)\} \Rightarrow \{(param, OUT), (time, morning)\}$ means that user John requests outgoing application services during the morning.
AU-PaD	Given a user (or a user category), in what period of the year does he request services and which service parameters are specified?	$\{user\} \Rightarrow \{param, date\}$	$\{(user, John)\} \Rightarrow \{(param, OUT), (date, winter)\}$ means that user John requests outgoing application services in winter.
RS-T	Given a user (or a user category), what service (class) does he request and in what time period?	$\{user\} \Rightarrow \{service, time\}$	$\{(user, John)\} \Rightarrow \{(service, CALL), (time, 2 - 6p.m.)\}$ means that user John requests the CALL service during the afternoon.
RS-D	Given a user (or a user category), what service (class) does he request and in what period of the year?	$\{user\} \Rightarrow \{service, date\}$	$\{(user, John)\} \Rightarrow \{(service, CALL), (date, December)\}$ means that user John requests the CALL service in December.
RS-P	Given a user (or a user category), what service (class) does he request and where?	$\{user\} \Rightarrow \{service, place\}$	$\{(user, John)\} \Rightarrow \{(service, CALL), (place, office)\}$ means that user John requests the CALL service in his office.
RS-Pa	Given a user (or a user category), what service (class) does he request and with which service parameters?	$\{user\} \Rightarrow \{service, param\}$	$\{(user, John)\} \Rightarrow \{(service, CALL), (param, OUT)\}$ means that user John requests the CALL service for outgoing calls.

(i.e., it includes a superset of the attributes of rule templates in Table 4.2), or (b) it references different attributes, thus representing knowledge that needs not to be separately classified. Table 4.3 defines templates for a subset of the specialized rules in category (a). Specialized templates are grouped in two subsets, analogously to templates in Table 4.2. The first six rule templates in Table 4.3 are specializations of the AU class template, while the last rules are specializations of *RS*. For instance, the specialized rule template $\{user\} \Rightarrow \{place, time\}$ adds the time attribute to the rule template $\{user\} \Rightarrow \{place\}$. Hence, it specializes the application usage class by also considering the correlation with the time period of the user requests.

All the categories in Tables 4.2 and 4.3 focus on different characteristics of the user-system interaction. User personalization has the goal of enhancing user-friendliness of the applications by capturing valuable recurrences in user habits. The knowledge provided by user rule analysis can be exploited

Table 4.4: CAS-MINE: length-2 service rule classes

Class	Question	Rule template	Example
ST	Given a service (or a class of services), at which time period is it requested?	$\{service\} \Rightarrow \{time\}$	$\{(service, WEATHER)\} \Rightarrow \{(time, morning)\}$ means that weather forecasts are requested in the morning.
SD	Given a service (or a class of services), in which period of the year is it requested?	$\{service\} \Rightarrow \{date\}$	$\{(service, WEATHER)\} \Rightarrow \{(date, June)\}$ means that weather forecasts are requested during June.
SP	Given a service (or a class of services), where is it requested?	$\{service\} \Rightarrow \{place\}$	$\{(service, CALL)\} \Rightarrow \{(place, office)\}$ means that the CALL service is requested in the office.
SPa	Given a service (or a class of services), which parameters are requested?	$\{service\} \Rightarrow \{params\}$	$\{(service, CALL)\} \Rightarrow \{(param, OUT)\}$ means that the CALL service is requested for outgoing calls.

to (i) select the first service (default service) to be suggested to connected user, (ii) automatically complete service parameters, (iii) plan future promotions, (iv) suggest the appropriate service type in a given context, and (v) automatically invoke a specific service when the user is in a given context. The proposed user rule classification is an effective tool to highlight hidden knowledge which may be relevant to this purpose. In Section 4.3.4 these templates are exploited for the analysis of real context datasets and the concrete usage of extracted rules is also discussed.

Service rules Service rules describe service characteristics, at any hierarchical level, regardless of the requesting users. These rules allow service providers to shape service provisioning to the current context. In service rules the attribute "service" always appears either in the body or in the head of the rule.

Similarly to user rules, service rules have been partitioned in more specific categories to support rule analysis. For each category, a set of interesting rule templates has been defined. Two significant subsets of rule templates for service rules of length 2 and 3 are summarized in Tables 4.4 and 4.5. They always include the service attribute in the body of the rule. Since interesting knowledge on service exploitation typically concerns its usage context (e.g., time or location), or the service parameters the user requires, four service templates of length 2 have been defined in Table 4.4. For a given service, the *ST* template defines the usage time, *SD* identifies the yearly period during which it is used, *SP* defines the usage location, while *SPa* identifies the service parameters. Table 4.5 reports a subset of the specializations of the rule templates in Table 4.4. For example, the specialized rule template $\{service\} \Rightarrow \{place, time\}$ adds either the place attribute to the *ST* rule template, or the time attribute to the *SP* rule template. Hence, it specializes the context, defined in terms of both place and time, in which a given service

Table 4.5: CAS-MINE: length-3 service rule classes

Class	Question	Rule template	Example
SPPa	Given a service (or a class of services), where is it requested and with which parameters?	$\{service\} \Rightarrow \{place, param\}$	$\{(service, WEATHER)\} \Rightarrow \{(place, home), (param, TODAY\_FORECAST)\}$ means that daily weather forecasts are requested at home.
SPT	Given a service (or a class of services), where is it requested and in which time period?	$\{service\} \Rightarrow \{place, time\}$	$\{(service, WEATHER)\} \Rightarrow \{(place, home), (time, evening)\}$ means that weather forecasts are requested at home in the evening.
SPD	Given a service (or a class of services), where is it requested and in which period of the year?	$\{service\} \Rightarrow \{place, date\}$	$\{(service, WEATHER)\} \Rightarrow \{(place, home), (date, summer)\}$ means that weather forecasts are requested at home in summer.
SPaD	Given a service (or a class of services), in what period of the year is it requested and with which parameters?	$\{service\} \Rightarrow \{param, date\}$	$\{(service, CALL)\} \Rightarrow \{(param, OUT), (date, week-end)\}$ means that outgoing calls are requested during the week-end.
SPaT	Given a service (or a class of services), in what time period is it requested and with which parameters?	$\{service\} \Rightarrow \{param, time\}$	$\{(service, CALL)\} \Rightarrow \{(param, OUT), (time, afternoon)\}$ means that outgoing calls are requested during the afternoon.
STD	Given a service (or a class of services), in what time and year periods is it requested?	$\{service\} \Rightarrow \{date, time\}$	$\{(service, CALL)\} \Rightarrow \{(date, winter), (time, afternoon)\}$ means that calls are requested in winter during the afternoon.

is frequently requested.

Service rules support service providers in shaping the offered services to the user needs. The extracted knowledge can be exploited by service providers to (i) size system resources and (ii) define a default profile for new users. The exploitation of service rule templates to effectively support these activities is discussed in Section 4.3.4, which reports several service rule examples discovered in real context datasets and discusses their usage.

4.3.4 Experimental evaluation of the CAS-MINE framework

The CAS-MINE framework has been evaluated by means of a large set of experiments addressing the following issues: (i) the characteristics of the mined knowledge (Section 4.3.4), (ii) the quality of the mined rules (Section 4.3.4), and (iii) the rule extraction performance (Section 4.5) in terms of execution time and number of mined rules.

All the experiments were performed on a 3.2 GHz Pentium IV system with 2 GB RAM, running Ubuntu 8.04. The CAS-MINE framework was implemented in the Python programming language [95].

Real context datasets

Three real context datasets, called *mDesktop*, *Recs*, and *TeamLife* were provided by Telecom Italia Lab. Regarding privacy concerns related to real context data usage, please notice that (i) experimental data were collected from voluntary users that provide their whole informed consent on personal data treatment for this research project and (ii) real user names were hidden throughout the paper to preserve identities.

mDesktop dataset. The Telecom Italia mobile desktop application provides different services to users (e.g., weather forecast) through mobile devices. The *mDesktop* application provides 26 different services. The mDesktop dataset contains 4487 records providing information on requested services and context (e.g., time and location, when available) of the logged users. The analyzed dataset includes the requests of 20 different (trial) users over a time period of one year.

To perform generalized rules mining, a taxonomy including the following generalization hierachies has been defined.

- date → month → trimester → year
- time stamp → hour → timeslot (two hours timeslots)→ day period (AM/PM)
- service → class of service
- latitude:longitude → city → country
- phone number → call type (PERSONAL/BUSINESS)

We also considered different generalization hierarchies for the time stamp attribute. In particular, we considered different time slots (e.g., four hours time slots and eight hours time slots), which provided similar analysis results.

Recs dataset. The *Recs* system provides recommendations to users on restaurants, museums, movies, and other entertainment activities. Each user can request a recommendation, enter a score, or update a score for an entertainment center. The *Recs* system provides these three services to the end users. The analyzed dataset was obtained by logging the requests of 20 users and their locations over a time period of three months. The dataset contains 5814 records. For the *Recs* dataset, the following generalization hierarchies have been considered:

- date → month → trimester → year
- time stamp → hour → timeslot (two hours timeslots)→ day period (AM/PM)
- latitude:longitude → city → country

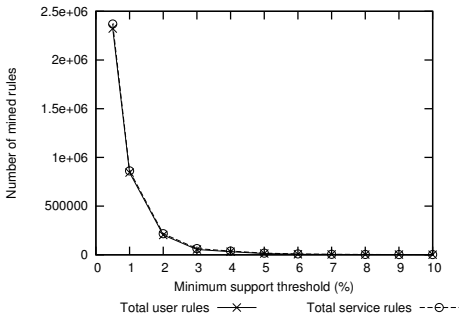
TeamLife dataset. The *TeamLife* dataset was generated by logging the activities of the users of the *TeamLife* system. The users of *TeamLife* can upload files, photos, or videos and share them with the other users. Four services are offered and the logged users of the *TeamLife* system are 20. Also this dataset includes context information, in particular time and location of the users. The dataset includes 1197 user requests collected over a time period of three months. For the *TeamLife* dataset the same taxonomy of the *Recs* dataset has been exploited.

Characteristics of the rules mined by CAS-MINE

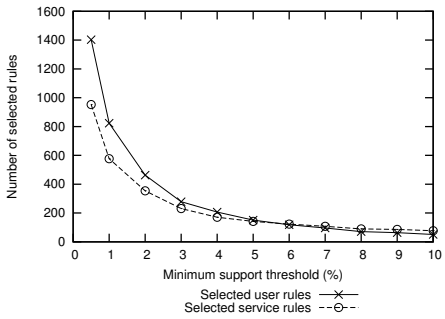
To characterize the rules discovered by CAS-MINE, we analyze the following issues: (i) The effect of the support threshold on the number of extracted patterns (Section 4.3.4), (ii) the impact of the generalization process (Section 4.3.4), and (iii) the rule distribution among templates (Section 4.3.4).

Effect of the support threshold Since the minimum support threshold enforced during the mining step significantly affects the number of extracted rules, in Figure 4.8 we report both (i) the number of user and service rules and (ii) the number of user and service rules selected by CAS-MINE (i.e., the rules belonging to the classes defined in Section 4.3.3). The analysis was performed without enforcing any minimum confidence threshold. Since the obtained results are comparable for all the three datasets, *Recs* is discussed as representative one (see Figure 4.8).

The number of mined rules significantly increases for minimum support values lower than 1% (see Figure 4.8(a)). Hence, it becomes difficult to exploit the extracted knowledge to create user and service profiles, since too many rules are available for each user and service. However, many rules either represent irrelevant information from an applicative point of view, or are (longer) specializations of other rules. By exploiting the user and service templates presented in Section 4.3.3, the number of selected rules (see Figure 4.8(b)) significantly decreases and becomes manageable. The number



(a) *Recs* dataset: number of user and service rules



(b) *Recs* dataset: number of user and service rules selected by CAS-MINE

Figure 4.8: CAS-MINE: number of user and service rules by varying the minimum support threshold. Minconf = 0%

of selected rules is up to three orders of magnitude smaller than the total number of extracted user and service rules for the *Recs* dataset (see Figure 4.8(b)). For the *TeamLife* and *mDesktop* datasets (detailed charts are not reported for lack of space) the number of selected rules is on average at least an order of magnitude smaller than the total number of extracted user and service rules, independently of the value of the minimum support threshold. Hence, CAS-MINE templates allow selecting a smaller set of rules that are interesting also from an applicative point of view. In particular, discarded rules are, for a large majority, specializations of other rules. A reduced number of rules include attribute combinations deemed as not relevant by the analysis of a domain expert. Some interesting applications of the selected rules are discussed in Section 4.3.4.

Impact of generalization Figure 4.9 shows, for different settings of minimum support and for all datasets, the percentage of rules including at least one generalized item on the set of rules extracted by CAS-MINE. For all datasets, the percentage of rules containing generalized items is at least equal to 70%.

Since in the CAS-MINE framework infrequent items are aggregated during the extraction process, the percentage of generalized rules increases when the support threshold is increased. When very high support thresholds (e.g., 10%) are enforced, most extracted rules include generalized items belonging to the top levels of the taxonomies (e.g., `location`: ITALY ⇒ `date`: 2008).

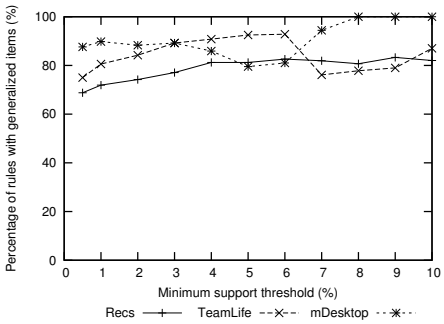


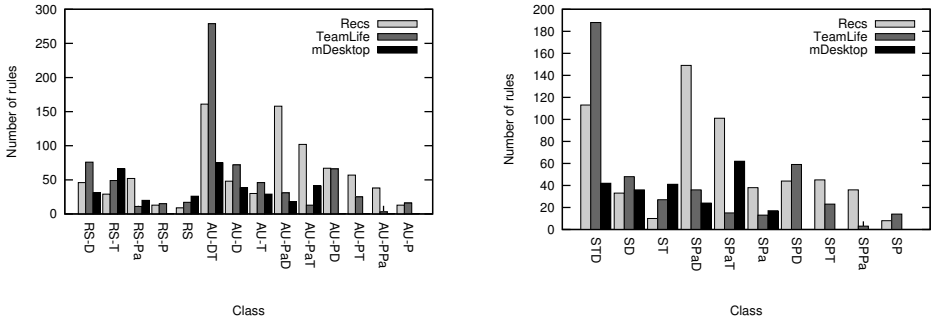
Figure 4.9: CAS-MINE: percentage of selected rules with generalized items by varying the minimum support threshold. Minconf = 0%

These rules are usually too general to provide interesting knowledge. Differently, when lower support thresholds are enforced (e.g., in the range 1%-4%), the extracted generalized rule set includes also non-top level elements of the taxonomy and more interesting knowledge is mined.

The extraction of generalized rules allows highlighting correlations that traditional association rules would hide because of their low support. The CAS-MINE approach allows a small set of low support association rules to be lazily aggregated into a higher support generalized association rule satisfying the support threshold.

Rule distribution among rule templates Extracted rules are analyzed by investigating how rules are spread among the classes defined in Section 4.3.3. Figure 4.10 shows, for all datasets, the number of extracted rules in each class with a minimum support threshold equal to 1%. Results for user rules are reported in Figure 4.10(a), while the ones obtained for service rules are reported in Figure 4.10(b). For the *Recs* and *TeamLife* datasets at least one rule is extracted for each class of user rules (see Figure 4.10(a)). Differently, for the *mDesktop* dataset some classes are empty. Since a very large number of user requests in the log file of the *mDesktop* application does not report the user location, the top level items in the place hierarchy are characterized by a support lower than 1%. Hence, rule templates characterized by the place attribute in the head are not populated.

For user rules (see Figure 4.10(a)), the class with the largest number of rules is *AU-DT*, characterized by template $\{user\} \Rightarrow \{date, time\}$. The cardinality of *AU-DT* is larger than that of its shorter versions *AU-D* and *AU-T*. This effect is due to the peculiar characteristics of the date and time attributes. In particular, both the date and time attributes (i) are always



(a) User rules

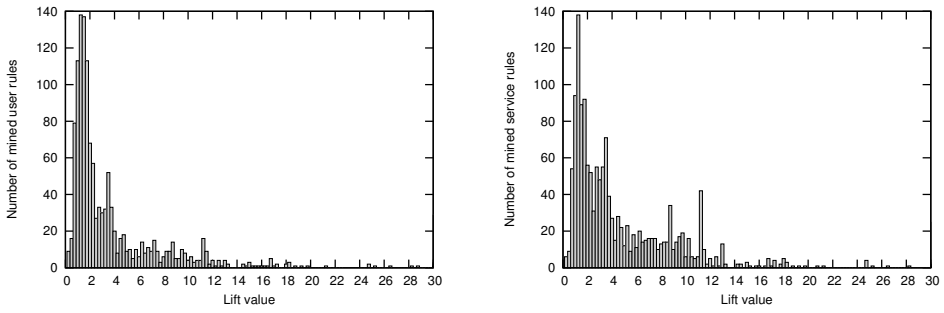
(b) Service rules

Figure 4.10: CAS-MINE: number of selected rules for each subclass of interest. Minsup=1%

specified together in the time stamp of the logged event, (ii) are characterized by a number of distinct values larger than the number of distinct values of the other attributes, and (iii) are characterized by hierarchies with four levels. Thus, the generalization process generates many relevant combinations of these attribute values, which yield the described behavior. In general, any attribute characterized by a large number of distinct values and a deep hierarchy (tree) may generate a large set of rules.

The number of extracted service rules for each dataset and class is reported in Figure 4.10(b). Similarly to user rules, also for service rules the templates including the date and time attributes are the most populated. In particular, the time and date combination (template *STD*) and the combinations with the param attribute (templates *SPaD* and *SPaT*) generate on average many rules.

Quality of the mined generalized rules The rule quality analysis is focused on defining interestingness measures able to represent both the significance and utility of the extracted knowledge. Different objective measures [107] (e.g., lift, Pearson correlation coefficient, Jaccard measure, Mutual information) can be exploited to rank the extracted patterns according to their degree of interest. Then, only high ranked rules are presented to the analyst. In CAS-MINE the lift measure is used to highlight the most interesting rules and is discussed in Section 4.3.4. The validation of the extracted knowledge, performed by domain experts, i.e., employees of Telecom Italia, is discussed in Section 4.3.4. The domain experts that manage



(a) *mDesktop* dataset: number of user rules

(b) *mDesktop* dataset: number of service rules

Figure 4.11: CAS-MINE: number of selected rules by varying the lift value. *mDesktop* dataset. Minsup = 1%, Minconf = 0%

the analyzed services acknowledged that the rules discovered by CAS-MINE represent valuable knowledge for both user and service profiling.

Ranking interesting rules by means of the lift measure As discussed in Section 4.3.3, the lift measure is used by CAS-MINE to rank mined rules and to highlight the most interesting ones. The distribution of lift values on rules extracted on *mDesktop* dataset is analyzed as a representative example. Figure 4.11 shows (i) the histogram of the number of mined user rules (Figure 4.11(a)) and (ii) the histogram of the number of mined service rules (Figure 4.11(b)) depending on their lift value. Rules were extracted by setting the minimum support threshold to 1% and without enforcing any confidence threshold (i.e., Minconf = 0%). Rule numbers for lift values greater than 30 are not reported to enhance readability of the plot. The numbers of user and service rules with lift greater than 30 are 43 and 40, respectively.

Figure 4.11 shows that the majority of the mined rules is positively correlated (i.e., lift value greater than 1) in both charts. Lift values greater than 10 highlight a reduced set of rules worth a careful inspection. Some of them will be discussed in Section 4.3.4. A limited percentage of extracted rules is negatively correlated (i.e., 7.9% of user rules and 4.6% of service rules). Also in this case, lift values below 0.5 highlight a small set of negatively correlated rules. Some of them are discussed in Section 4.3.4.

Domain expert validation

The extracted rules have been analyzed by domain experts to assess the effectiveness of the CAS-MINE framework in discovering interesting and useful knowledge. The domain experts suggested possible usage scenarios for context-aware user and service profiling.

Habits of specific users (or user categories). The habits of users may be characterized by some kind of recurrence. For example, the following rules allow to discover valuable knowledge about a generic user of the *mDesktop* application, named *Rossi*[2]. They have been mined by enforcing a support threshold equal to 1% (i.e., absolute threshold=45).

1. Class RS
   (a) `user`: Rossi ⇒ `service`: CALL ($sup = 1.3\%, conf = 53\%, lift = 41.5$)
   (b) `user`: Rossi ⇒ `service`: SMS ($sup = 1.1\%, conf = 47\%, lift = 41.5$)
2. Class $AU - T$
   (a) `user`: Rossi ⇒ `hour`: PM ($sup = 2.3\%, conf = 94\%, lift = 25.4$)

The first two above rules belong to the RS rule template. They highlight that user *Rossi* is interested in two specific services, CALL and SMS, with rule confidence close to 50%. They provide relevant knowledge on this user attitudes. If a larger support threshold is enforced, for instance 2% (absolute threshold = 90), the following rule is extracted.

- `user`: Rossi ⇒ `service`: Communication ($sup = 2.4\%, conf = 100\%, lift = 22.8$)

This rule, with confidence 100%, is a high level grouping of the two former rules. It shows that *Rossi* is exclusively interested in the *Communication* service superset, which groups the CALL and SMS services.

Rule templates, described in Section 4.3.3, also allow characterizing different user habits. They entail (i) The service type users are mainly interested

[2]Due to privacy concerns actual individual names are not provided.

in, (ii) the context in which requests are commonly submitted, and (iii) the parameters that are frequently used. For instance, rule 2(a) highlights an intensive system usage by user *Rossi* during the afternoon/evening (confidence 94%).

Profiling user habits by exploiting negative correlation. The following rule, discovered in the *TeamLife* dataset, belongs to class *RS*, but, differently from previous rules, it is characterized by negative correlation (lift lower than 1).

- `user`: Verdi ⇒ `service`: PHOTO ($sup = 1.3\%, conf = 12\%, lift = 0.16$)

This rule shows that the user *Verdi* frequently uses the PHOTO service (the support of the rule is 1.3%). However, since the lift value is close to 0, it means that *Verdi* uses the PHOTO service less than expected. A user specific marketing action may target *Verdi* to promote the PHOTO service.

This information can also be exploited for cross selling purposes. Given a frequently used service, e.g., the FILE service in the positively correlated rule below

- `user`: Verdi ⇒ `service`: FILE ($sup = 9.94\%, conf = 86.9\%, lift = 6.30$)

a cross selling marketing action could consider several services belonging to the same aggregate group and select, as a service to be promoted, a negatively correlated service in the same group. Advanced knowledge on user habits can thus be exploited to generate promotions of (similar) rarely requested services (e.g., the PHOTO service for user *Verdi*).

Profiling services. Service rules highlight frequently used services and frequently asked parameters for each service, independently of the specific user who submits the requests. In the *mDesktop* dataset, by enforcing a minimum support threshold equal to 1% (absolute threshold = 45), the following generalized rules are extracted.

1. class SD

- `service`: TWITTER ⇒ `month`: February ($sup = 2.9\%, conf = 31\%, lift = 1.9$)

2. class SPa
   - `service`: CALL ⇒ `inout`: OUT ($sup = 1.1\%, conf = 89\%, lift = 48.9$)
3. class SP
   - `service`: TLWIDGET ⇒ `location`: Turin ($sup = 1.2\%, conf = 50\%, lift = 5.0$)

The first rule, belonging to the *SD* class, describes the period of usage of a specific service. Similar rules can be exploited to suggest default services, which could be temporally updated during the year depending on the services expected to be interesting in the considered period. The second rule, belonging to the *SPa* class, highlights the correlation between a service type and its parameters. In this case, call services are mainly exploited to perform outgoing calls. Finally, the last rule, belonging to class *SP*, describes the location in which a specific remote service is commonly used. This information could be useful both to size the provider system and to promote services in particular cities or regions.

4.4 A social application: the TweCoM Framework

The TweCoM (Tweet Context Miner) framework is a data mining environment, based on the GenIO algorithm, that addresses data mining and knowledge discovery from Twitter user-generated content and the relative context effectively. The main architectural blocks of the TweCoM framework are shown in Figure 4.12. A brief description of each block follows.

Representation of user-generated content. The most relevant Twitter posts (i.e., tweets) are retrieved by means of an ad-hoc crawler. Each tweet is represented as a record (i.e., set of items) which describes both content features (e.g., the most relevant keywords) and contextual features (e.g., the geographical location).

Taxonomy generation. This block addresses the automatic generation of taxonomies over content and contextual data items. Depending on the kind of

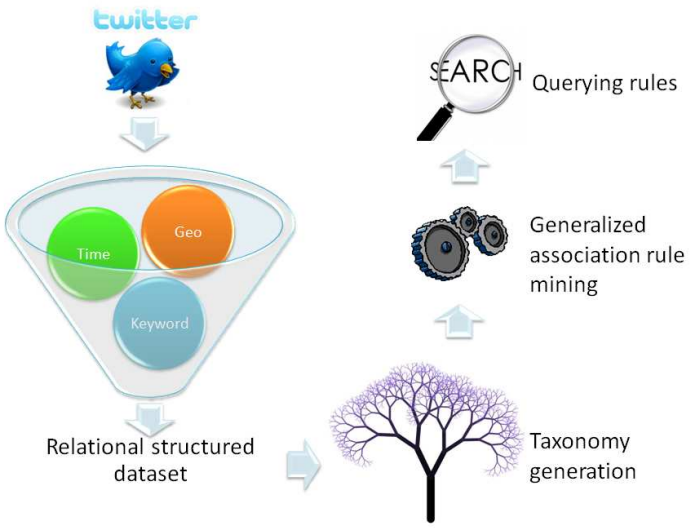


Figure 4.12: The TweCoM framework architecture

information, different approaches are exploited for the taxonomy generation. Taxonomies provide a high level abstraction of the mined knowledge which allows representing tweet features at different levels of abstraction.

Generalized association rule mining. Generalized association rule mining discovers relevant high level recurrences hidden in the tweet collection. The generalization process is driven by the previously generated taxonomies. The extraction of generalized association rules is performed by means of a two-step process: (i) frequent generalized itemset extraction and (ii) rule generation from the extracted frequent itemsets. The first mining step is based on the GENIO algorithm (see Chapter 4).

Querying rules. The extracted rules are queried, based on either their content or their schema, to more efficiently retrieve the information of interest. The resulting rules are ranked based on their confidence and support values to better support in-depth analysis.

A more detailed description of each block and its functionalities is presented in the following sections.

4.4.1 Representation of user-generated content

Twitter (`http://twitter.com`) is one of the most popular microblogging and social networking service. The Web service is mainly based on the messages,

```
2 - UserA: {results: [{profile_image_url:..., created_at: Sun, 10 Oct 2010
12:43:31 +0000, from_user:.., metadata: {result_type:recent}, to_user_id: X,
text: This is a text message, id: Y, from_user_id: X, to_user: UserB,
geo:{coordinates:+X -Y id: Z, place: New York City, place_type: city
Country: NY-United States of America}, iso_language_code: en, source..

2 - UserB: {results: [{profile_image_url:..., created_at: Wed, 20 Oct 2010
13:30:12 +0000, from_user:.., metadata:{result_type: recent}, to_user_id: X,
text: This is another message, id: X, from_user_id: X, to_user: User2,
geo:{coordinates: +X -Y id: Z, place: Los Angeles, place_type: city
Country: California-United States of America}, iso_language_code:en, source..
```

Figure 4.13: A simplified example of tweets in the JSON data format

named tweets, posted by users. Tweets are posts, of at most 140 characters, that are publicly visible by default. The user-generated content (i.e., tweets) can be accessed by means of Search Application Programming Interfaces (APIs) provided by the social network site. Data returned by Twitter APIs is stored in the JSON format (Java Script Object Notation), which is an XML-based standard for client-server data exchange. A simplified example of two tweets tailored to the JSON format is reported in Figure 4.13.

As shown in Figure 4.13, tweets are characterized by a short text message enriched with several context pieces of information (e.g., source location coordinates, city, date, hour). Some contextual features are peculiar characteristics of the context in which tweets are posted by users (e.g., the source location coordinates), while others are just high level aggregations of the previous ones (e.g., the city). Consider the text message and low level contextual features first. Couples $(attribute, value)$, where *attribute* is the text message or the description of the context feature (e.g., the date) and *value* is the collected information (e.g., "This is a text message", $2010 - 10 - 10$), are denoted as items in the following. In the case of continuous attributes, the value range is discretized into intervals and the intervals are mapped to consecutive positive integers.

Since data retrieved by Twitter is not compliant with a relational structured dataset, a preprocessing phase is needed. During the joining process, data are tailored to a common relational data format by means of a data cleaning process. Data cleaning also discards useless and redundant information and correctly manages missing values.

For each tweet, the following information are extracted:

- geographical information
- tweet publication date and time stamp

TWEETS
((**Tweet ID, 1**), (Propagation level,2), (Username, UserA), (Place, New York City), (Date, 2010-10-10), (Time, 12:43:31 +000), (Text, key_{1a} key_{1b}))
((**Tweet ID, 2**), (Propagation level,2), (Username, UserB), (Place, Los Angeles), (Date, 2010-10-20), (Time, 13:30:12 +000), (Text, key_{2a} key_{2b}))

Figure 4.14: A simplified structured dataset generated from tweets

- response/citation level of propagation
- tweet keywords

While the first two tweet features can be directly extracted from the JSON data format, the tweet propagation level and keywords are usually not defined. Therefore, we record the tweet response/citation propagation level during data crawling and process the collection to select the of most representative words within each tweet content. Tweet contents are represented by means of the bag-of-word (BOW) representation in which stopwords, numbers, and website URLs are removed to avoid noisy information and on which the Porter stemming algorithm [93] is applied. The BOW representation is associated with the statistical measure of the term frequency-inverse document frequency (tf-idf) that evaluates the relevance of a word in the whole collection. The representative tweet keywords are thus defined as the top-k words with highest tf-idf values.

After preprocessing, the collected information can be modeled as a structured dataset, where records represent Twitter messages (i.e., tweets) by means of (i) their most representative tweet keywords, and (ii) their related contextual features. This representation will be exploited to effectively address generalized association rule mining from tweet collections. An example of structured context dataset record obtained by including the content of a portion of two Twitter messages, presented in Figure 4.13, is reported in Figure 4.14. The primary key (Tweet ID attribute) is in boldface.

Tweet crawler

Twitter APIs are general-purpose libraries that allow the efficient retrieval of tweets from the Web. However, a procedure to extract a collection of tweets and their corresponding relationships (e.g citations or responses) is

Algorithm 2 Tweet crawler

Input: k /*number of top tweets*/, $keys$ /*set of keywords*/, d /*search maximum depth*/
Output: T /*collection of tweets*/
1: $T = \emptyset$, $i = 0$
2: // initialization of the source set
3: **if** $keys = \emptyset$ **then**
4: S = set of top-k toptweets
5: **else**
6: S = set of top-k tweets retrieved by means of keyword search using $keys$
7: **end if**
8: **repeat**
9: $C = \emptyset$ // candidate set initialization
10: **for all** s in S **do**
11: $C = C \cup \{c_i$ tweets directly linked to $s | c \notin \{S \cup T\}\}$
12: **end for**
13: $T = T \cup S$ // update of tweet collection
14: $S = C$ // update of source set
15: $i = i + 1$
16: **until** $i \neq d$
17: $T = T \cup S$ // it includes the tweets in S in the final collection set T
18: **return** T

not provided yet. Thus, a tweet crawler to effectively query Twitter Web service and retrieve a collection of linked tweets is needed. To this aim, the following two sets of tweets are defined: (i) the source S and (ii) the collection T sets. The following parameters are provided to the crawler: (i) the number of top-k tweets that will be retrieved during the first step of the procedure, and (ii) an optional set of keywords $keys$. If the $keys$ set is empty the source set is initialized with the tweets belonging to the toptweet category, else with the top-k results retrieved by means of a keyword search (lines 3-7). For each tweet belonging to the source set S, all the tweets that are directly linked to the original post (i.e., answers and/or citations) and are not just visited by the procedure are inserted in a candidate set C (lines 10-12). Then, the original posts in the source set S are moved into the collection set T (line 13), while the candidate set becomes the new source set (line 14). This step is recursively repeated for each tweet belonging to the source set until a termination condition is satisfied. When the stop condition is reached, all the tweets belonging to the source set are moved into the collection set. Different termination conditions can be employed. To investigate tweet propagation, we defined a maximum depth level d of responses/citations as crawler stop condition (see line 16). Since Twitter system enforces some upper bound constraints for tweet download by means of APIs, an approach, similar to TPC congestion control, is adopted to query the Twitter APIs efficiently.

4.4.2 Taxonomy generation

A taxonomy is a hierarchical knowledge representation that defines is-a relationships between concepts and their instances (i.e., the items). Taxonomies are used to classify objects, define semantic relationships between concepts and provide additional information on data. Most of the previously proposed generalized rule mining approaches (e.g., [52, 103]) commonly rely on user-provided taxonomies. Moreover, these taxonomies are usually general-purpose, thus they may not reflect the real meaning of the information stored in the data collection. The automatic extraction of taxonomies from data items in a relational dataset is definitely a challenging task. Since Twitter APIs provide contextual knowledge as "flat" attributes (i.e., relationships among data objects are not provided yet), this block automatically generates a taxonomy over a selection of tweet record attributes.

Context taxonomy generation

Taxonomies over contextual features (e.g, spatial and temporal information) can be derived by means of aggregation functions based on a hierarchical model. The hierarchical model represents the relationships between different levels of aggregation. Similarly to what usually done in data warehousing, these pieces of information are extracted by means of Extraction, Transformation and Load (ETL) processes, named here aggregation functions, defined by the user. For example, in the relational tweet representation, aggregations functions may define either associations among different context attributes (e.g., $City \Rightarrow State$) or aggregations over a singular context attributes (e.g., $Date \Rightarrow Semester$) which could be derived by simply parsing the corresponding attribute domain values.

Given a set of aggregation functions among UGC features, the taxonomy generation block allows automatically building a taxonomy. It associates with each item the corresponding set of generalizations organized in a hierarchical fashion. For instance, consider a temporal context feature included in the UGC (e.g., $Month$) that represents high level knowledge abstraction of another context feature (e.g., $Date$). A conceptual hierarchy of aggregations may be devised by mapping the two attribute domains by means of the corresponding aggregation function (e.g., $Date \Rightarrow Month$). Consider again the $Date$ attribute and its high level aggregation $Semester$. Although the corresponding higher level attribute does not exist yet, the corresponding mapping may be simply derived by parsing the lower level $Date$ domain values (e.g., $2010 - 10 - 10$) and generating upper level concepts

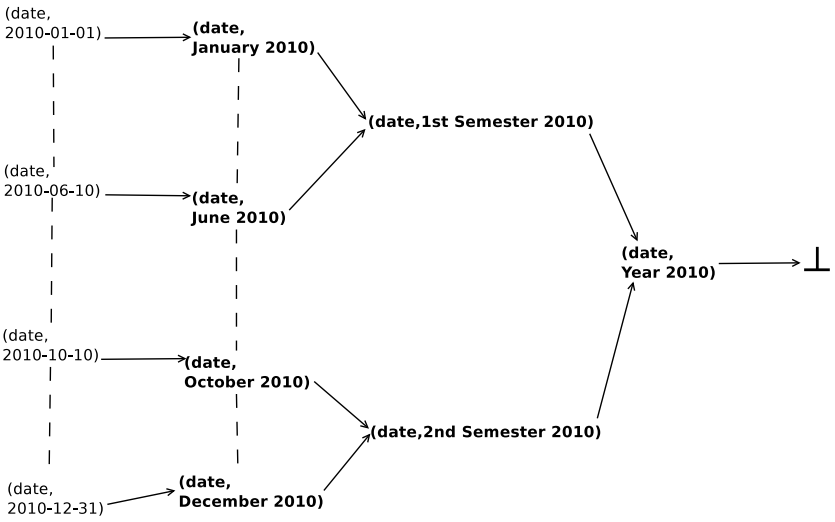


Figure 4.15: A portion of the generalization hierarchy for the date attribute

(e.g., 2*nd Semester* 2010) according to the corresponding aggregation function (i.e., $Date \Rightarrow Semester$). An example of taxonomy built over the *date* attribute is reported in Figure 4.15.

In Table 4.6 the aggregation functions exploited for the automatic generation of the taxonomy over temporal and spatial data features are resumed. However, the framework allows the usage of different aggregation functions as well.

Keyword taxonomy generation

In literature several hierarchical organizations of concepts have been proposed in textual data analysis (e.g., [11, 69]). This block focuses on combining both content and contextual high level data features to generate a taxonomy over tweet content keywords as well. The generation of a taxonomy directly from the user-generated content (UGC) collection may provide additional knowledge about the content and the relationships between words/concepts that are of major interest for the social community. In the following, this issue is addressed by adopting a clustering-based approach.

As discussed in Section 4.4.1, a tweet content collection can be represented by means of a tf-idf matrix. For each tweet content, both the Porter stemming

Data feature	**Aggregation function**
Temporal	$Date \Rightarrow WeekDay$ $Date \Rightarrow Month$ $Month \Rightarrow Year$ $Time \Rightarrow Hour$ $Hour \Rightarrow TimeSlot$
Spatial	$GPSCoordinates \Rightarrow Id$ $Id \Rightarrow Place$ $Place \Rightarrow Region$ $Region \Rightarrow State$

Table 4.6: TweCoM: examples of spatial and temporal aggregation functions

algorithm [93], which reduces words belonging to the collection to their base or root form, and a filter on stopwords, numbers, and webpage URLs, to remove redundant and noisy information, are applied.

The tweet content collection can be represented in a matricial form TC in which each row represents a stemmed word (i.e., a term) of the collection while each column corresponds to the content of a tweet. Each element of the matrix TC is denoted as the tf-idf (Term Frequency - Inverse Document Frequency) value for a term. It is computed as follows:

$$tc_{ij} = \frac{n_{ij}}{\sum_{k \in \{q \,:\, t_q \in tw_j\}} n_{kj}} \cdot \log \frac{|TW|}{|\{tw \in TW \;:\; t_i \in tw\}|} \tag{4.2}$$

where n_{ij} is the number of occurrences of i-th term t_i in the j-th tweet content tw_j, TW is the collection of tweet contents, $\sum_{k \in \{q \,:\, t_q \in tw_j\}} n_{kj}$ is the sum of the number of occurrences of all terms in the j-th tweet content tw_j, and $\log \frac{|TW|}{|\{tw \in TW \,:\, t_i \in tw\}|}$ represents the inverse document frequency of the i-th term t_i (i.e., the logarithm of the ratio between the tweet collection cardinality and the number of tweets whose content includes term t_i).

Since a taxonomy has a hierarchical structure, a hierarchical clustering approach is applied to the tf-idf matricial representation of the tweet content collection. The hierarchical clustering approaches build a tree-based structure, called dendrogram, that represents how the tweet contents are grouped together at different levels of aggregation. Despite the algorithm may support different approaches and similarity measures to build the dendrogram structure, a centroid linkage approach based on the cosine similarity measure is adopted. Cosine similarity measure is one the mostly used measures

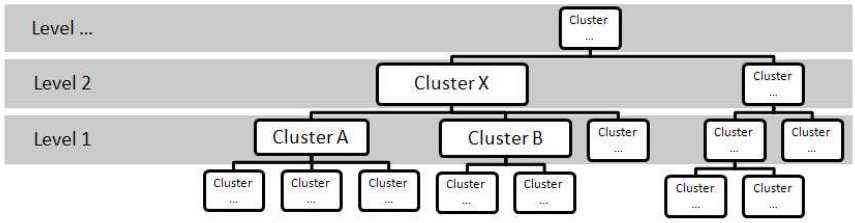


Figure 4.16: An example of dendrogram cut for keyword taxonomy generation

to compute similarity between documents in the BOW representation. The similarity between two tweet contents/clusters, represented by means of their corresponding tf-idf vectors A and B, is computed as the dot product between the two vectors normalized by the product of their magnitude. The employed clustering approach assigns a vector to each cluster which represents the cluster itself. It is computed as the average between all the tweets belonging to the cluster. Thus, the centroid provides the set of the the most representative (i.e., highest tf-idf values) keywords of the tweet group.

The adopted approach exploits the dendrogram structure and the centroid representation to automatically generate a taxonomy over the tweet text content. The method cuts the hierarchical clustering scheme at different levels l according to the analyst request. For each cluster C_l at level l, the representative keywords of the tweets are the top-t terms with the highest tf-idf values. Since a cluster at level $l+1$ is composed of two or more clusters belonging to the level l, keyword aggregation functions, formally defined in Definition 18, is exploited to construct a taxonomy over tweet content keywords.

Definition 18 Keyword aggregation function. *Let t_l and t_{l+1} be the sets of keywords representing two clusters C_l and C_{l+1} obtained by two different cuts of dendrogram (i.e., two different levels). If C_l is a subset of C_{l+1} (i.e., $C_i \subseteq C_j$), the aggregation function between two keywords $key_l \in t_l$ and $key_{l+1} \in t_{l+1}$ is: $key_l \Rightarrow key_{l+1}$.*

In the following an example of the taxonomy generation process over the tweet content keywords is reported. Consider the dendrogram shown in Figure 4.16. Suppose that the analyst decides to cut the dendrogram between the levels 1 and 2. At level 1, clusters A and B are characterized by the keywords "Obama" and "Bush" respectively. At level 2 the two clusters

are merged in cluster X that is represented by keyword "President", i.e., the term with highest tf-idf value among all tweet contents belonging to X. Thus, the taxonomy generation process, according to the Definition 18, exploits the following keyword aggregation functions: $Obama \Rightarrow President$ and $Bush \Rightarrow President$.

4.4.3 Generalized association rule mining

Correlations hidden in a dataset D may be effectively represented by means of association rules [2].

The generalized rule mining block takes in input the structured dataset built over tweets, the generated taxonomies, and, possibly, some mining constraints (e.g,, minimum support and confidence thresholds). It discovers all the generalized association rules that satisfy the enforced constraints.

The mining task follows the usual two-step approach [103]: (i) extraction of the frequent generalized itemsets, and (ii) generation of the corresponding generalized rules. Since the first step is considered the most computationally intensive step [2], plenty of algorithms (e.g., Cumulate [103], GenIO [16], ML_T2LA-C [52]) have been proposed to efficiently perform this task. To perform the generalized itemset extraction task effectively, the GenIO algorithm [16] is exploited, while the generalized rule generation is performed by exploiting our implementation of the Apriori rule mining algorithm [103].

4.4.4 Querying rules

The block entails the selection and ranking of most valuable rules for better supporting in-depth analysis. Rule selection is constrained by either (i) the rule schema (i.e., the attributes that have to appear in the rule body or head), or (i) some specific rule items of interest. An example of schema constraint may be: $(Keyword, *) \rightarrow (Place, *)$. It selects all 2-length rules that include, respectively, an item characterized by attribute $Keyword$ in the rule body and attribute $Place$ in the rule head. For instance, the generalized rule $(Keyword, Sport) \rightarrow (Place, U.S.))$ satisfies the above schema constraint. Differently, an example of item constraint is: $\{*\} \rightarrow \{(Place, U.S.)$. It selects all rules that contain item $(Place, U.S)$ as rule consequent. Rule $(Keyword, Sport) \rightarrow (Place, U.S.))$ also satisfies the proposed item constraint.

Results of rule querying are sorted by confidence and support quality indexes to better support in-depth analysis. Although confidence and support are the most popular rule quality indexes [2], the TweCoM framework allows to easily integrate different quality indexes as well (e.g., lift [107]).

4.4.5 Experimental evaluation of the TweCoM framework

The efficiency and the effectiveness of the proposed framework have been evaluated by addressing the following issues: (i) the efficiency of the tweet crawler, (ii) the effectiveness of the adopted taxonomy inference procedure, and (ii) the propagation analysis of tweet answers and/or citations by means of generalized rule mining.

Data retrieval

The TweCoM framework exploits a crawler to effectively retrieve tweets from the Web. The retrieval procedure, described in Section 4.4.1, is exploited to follow the sequence of responses/citations to the top-k results of a keyword search until a maximum search depth is reached. A campaign of keyword searches is performed starting from a set of commonly used American terms and famous names (e.g., *Soccer*, *Obama*, *Society*). Searches are performed with the aim of discovering most significant trends in UGC generation and propagation. Examples of use-cases for the proposed TweCoM framework that concern the collected tweets are deeply discussed in the following. The impact of the following parameters on both the number of extracted tweets and the time spent in tweet retrieval is investigated: (i) the number of top results of the keyword search k, and (ii) the maximum search depth d. In Table 4.7 the number of tweets retrieved for the most relevant keyword searches is reported as well as the corresponding time elapsed in tweet crawling by setting the maximum search depth D=4 and by varying the the number of retrieved top results from 4 to 20. Both the elapsed time and the cardinality of the tweet set scale roughly linearly with the number of top results as long as the cardinality of different chains of answers/citations related to different top results are similar.

Keyword	$k = 4$		$k = 10$		$k = 15$		$k = 20$	
	Num. tweets	Time (s)	Tweets tweets	Time (s)	Num. tweets	Time (s)	Num. tweets	Time (s)
Obama	19,242	15.2	25,342	20.0	27,291	24.2	34,321	28.4
Clinton	16,543	13.3	18,765	15.3	21,454	18.2	24,652	22.9
Society	14,532	12.7	16,119	14.1	19,665	16.8	23,552	20.0
Health	11,683	10.7	12,753	12.1	14,664	15.7	18,421	18.1
War	9,332	8.6	11,356	10.2	13,568	12.7	16,742	15.9
Sport	12,687	12.3	14,331	15.1	17,876	17.1	19,864	19.5
Soccer	18,238	15.1	19,594	17.0	21,303	18.9	23,643	20.0
Los Angeles Galaxy	12,687	11.3	14,223	13.2	17,352	15.8	18,992	18.0
Stadium	9,623	9.8	11,200	11.1	12,782	12.9	14,782	14.8

Table 4.7: TweCoM: tweet set cardinality and elapsed time in tweet crawling by varying the number of retrieved top results. $d = 4$

Keyword	$d = 3$		$d = 4$		$d = 5$		$d = 6$	
	Num. tweets	Time (s)	Tweets tweets	Time (s)	Num. tweets	Time (s)	Num. tweets	Time (s)
Obama	21,112	16.1	25,342	20.0	29,345	28.6	65,345	47.7
Clinton	16,543	13.3	18,765	15.3	25,876	25.1	56,323	42.2
Society	14,112	11.2	16,119	14.1	23,453	23.1	49,075	36.0
Health	10,301	10.9	12,753	12.1	26,360	25.9	68,102	45.1
War	9,993	7.2	11,356	10.2	19,342	18.4	34,795	39.1
Sport	11,633	11.5	14,331	15.1	22,098	26.2	50,061	40.0
Soccer	15,114	15.1	19,594	17.0	28,026	27.0	52,037	44.5
Los Angeles Galaxy	12,996	11.4	14,223	13.2	21,753	19.1	42,664	39.0
Stadium	9,453	9.1	11,200	11.1	18,553	17.9	44,118	39.3

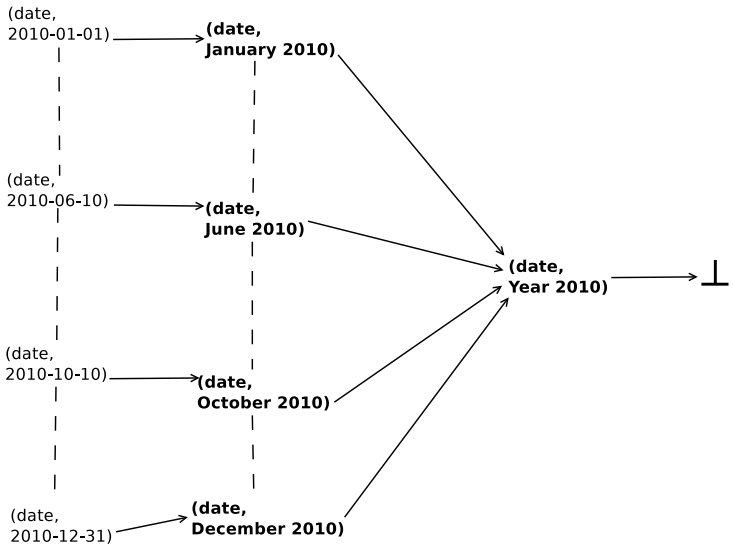
Table 4.8: TweCoM: tweet set cardinality and elapsed time in tweet crawling by varying the number of top results. $k = 10$

In Table 4.8 the number of tweets retrieved for the most relevant keyword searches is reported as well as the corresponding time elapsed in tweet crawling by setting the number of top results $k = 10$ and by varying the maximum search depth from 3 to 6. Both the elapsed time and the cardinality of the gathered tweet set scale more than linearly with the maximum depth search because of the combinatorial growth of the number of considered answers/citations. We denoted the setting $k = 10$ and $d = 4$ as standard configuration in the rest of the experimental section.

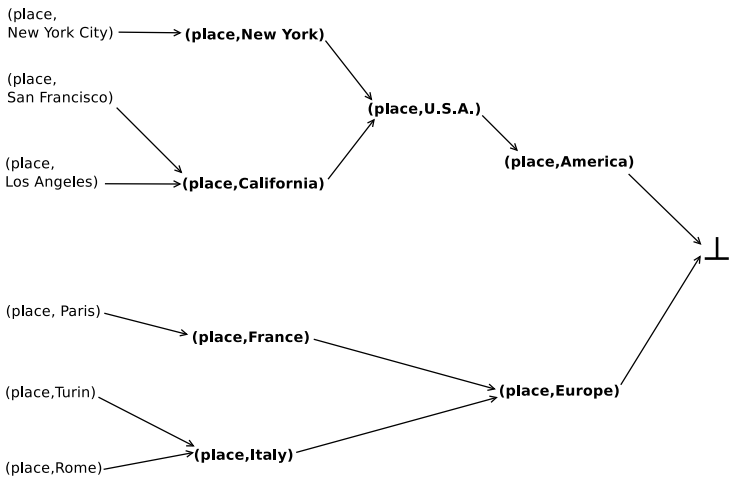
Taxonomy inference

Depending on the kind of processed information, the taxonomy generation procedure exploits either semantic relationships among tweet contextual features, or hierarchical clustering over Twitter textual message content. The analysis of the tweets retrieved by means of the tweet crawler is performed by setting its standard configuration (see Section 4.4.5). By exploiting the aggregation relationships reported in Table 4.6 over geographical and temporal tweet contextual features, a set of generalization hierarchies is built. A portion of the inferred taxonomy is reported in Figure 4.17.

The hierarchical clustering has been exploited to generate a taxonomy over tweet content keywords. For each cluster the keyword characterized



(a) Date attribute.



(b) Place attribute.

Figure 4.17: A portion of the generalization hierarchies for the date and place attributes

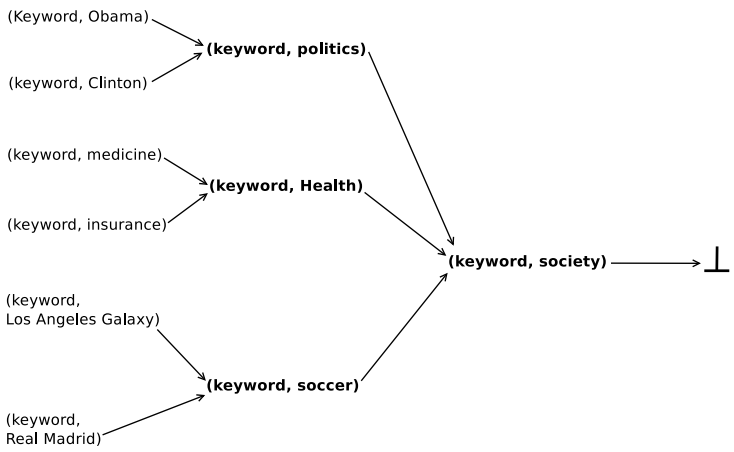


Figure 4.18: A portion of the generalization hierarchy for the tweet content keywords

by the highest tf-idf value is selected as representative (i.e., $t = 1$). To capture soccer teams, their names in the BOW representation is introduced. A portion of the resulting taxonomy built over tweet content keywords is reported in Figure 4.18. The devised aggregations are deemed relevant by domain experts. Thus, they are employed in the generalized rule mining process. The selected aggregation relationships between tweet contextual features (e.g., between *Los Angeles*, *California*, and *U.S.A.*) and content keywords (e.g., between *Soccer* and *Los Angeles Galaxy*) will be exploited in Section 4.4.5 to effectively characterize tweet propagation.

The distribution of the tf-idf weight among tweet contents is also addressed. In particular, in Figure 4.19 the tf-idf distribution of tweet collection concerning Obama is reported. Since tweets are characterized by a few number of words due to the post length limitation imposed by Twitter (i.e., at most 140 characters), the inverse document frequency impact becomes dominant. In fact, term frequencies are usually close to one and their distributions follow a power law. The inverse document frequency allows normalizing this distribution by assigning a score to the most relevant words in the tweet collection.

The performance of the hierarchical clustering algorithm employed in our approach is compared with the ones of the traditional k-means algorithm [83]. The cosine distance was used as similarity measure for both algorithms. The overall average silhouette achieved by the previously described tweet collections is measured. In the reported experiments, the dendrogram is cut at the levels at which the overall average silhouette is still higher than 0.8. In

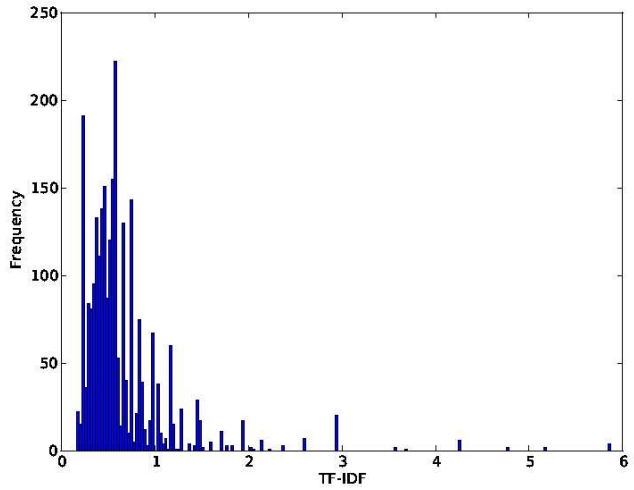


Figure 4.19: TweCoM: tf-idf distribution in the Obama tweet collection

Figure 4.20, the variation of the silhouette coefficient values yielded by the two algorithms in terms of the number of generated clusters is reported for one tweet representative collection. Similar results were obtained for the others. The hierarchical algorithm outperformed k-means for any number of generated clusters. Moreover, the silhouette for the hierarchical algorithm was averagely equal 0.77, while for the k-means was -0.08. Thus, the hierarchical approach could more efficiently identify well-formed clusters than the k-means approach.

Propagation analysis of tweet content

In this section, the effectiveness of the TweCoM framework in discovering valuable correlations from Twitter user-generated content is investigated. To address this issue, three different examples of use-cases for TweCoM are analyzed separately.

Use-case 1: Spatial propagation analysis of tweet content. This application scenario drives the analyst to retrieve and select the most relevant rules involving spatial information. To achieve this goal, the analyst performed the following steps: (i) post retrieval and categorization based on their propagation level (in the chain of answers/citations), (ii) generalized rule mining from each generated tweet set, and (iii) rule query-

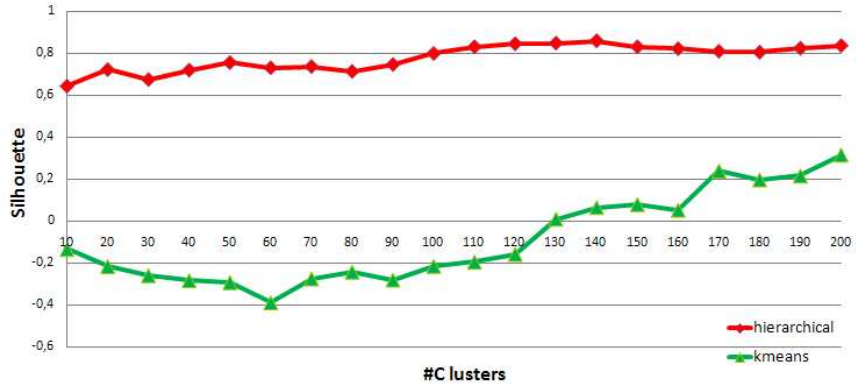


Figure 4.20: TweCoM: silhouette coefficient trend in a tweet collection

ing based on user-specified constraints involving spatial information (e.g., $(Keyword, *) \rightarrow (Location, *)$). The generalized rule mining process is performed by exploiting both (i) the tweets retrieved by the tweet crawler with the standard configuration, and (ii) the taxonomy reported in Section 4.4.5. During the generalized rule mining, the analyst enforced a minimum support threshold equal to 0.5% and a minimum confidence threshold equal to 50%. The analyst sorted rules according to, respectively, their confidence and support values to select the strongest and most recurrent patterns. The selection task is first performed by enforcing the schema constraint $(Keyword, *) \rightarrow (Location, *)$ over the set of mined patterns. From the set of tweets characterized by propagation level 1 (i.e., direct answers/citations of the top-k tweets), the following rules have been extracted:

Propagation Level = 1

(i) (Keyword, Los Angeles Galaxy) → (Location, Los Angeles) ($sup = 1.0\%$, $conf$=87%)

(ii) (Keyword, Los Angeles Galaxy) → (Location, San Francisco) (sup = 0.7%, $conf$=74%)

They state that the American soccer team name *Los Angeles Galaxy* frequently occurs in the content of tweets posted from Los Angeles and San Francisco. The analysis of the next propagation level (2) highlighted the following generalized rule:

Propagation Level = 2

(iii) (Keyword, Soccer) → (Location, Los Angeles) (sup = 1.5%, $conf$ = 87%)

The above rule is a generalization of the former ones. It states that a significant part of tweets whose content includes keyword *Soccer* have been posted from Los Angeles, where one of the most famous American soccer teams (i.e., the Los Angeles Galaxy) plays. At this level, patterns that involve singular teams became infrequent and, thus, they have not been extracted. However, due to the support-driven generalization process exploited by GenIO [16], their generalization is triggered over the taxonomy (see Figure 4.18). Indeed, singular soccer team names are generalized in their upper level aggregation *Soccer*. Among the all possible generalizations, one of them (i.e., rule (iii)) becomes frequent with respect to the minimum support threshold. The discovered knowledge prompted the analyst to deepen the investigation into some selected keywords (e.g., *Soccer*). Spatial propagation of tweets whose content includes keyword *Soccer* has been investigated by enforcing the following item and schema constraint $(Keyword, Soccer) \rightarrow (Location, *)$ on the set of mined rules. Most of the answer/citation authors were located nearby at submission time, as shown by the following rules extracted from tweets characterized, respectively, by propagation level 3 and 4.

Propagation Level = 3

(iv) (Keyword, Soccer) → (Location, California) ($sup = 0.9\%$, $conf = 71\%$)

Propagation Level = 4

(v) (Keyword, Soccer) → (Location, California) ($sup = 0.6\%$, $conf = 76\%$)

Twitter users that were interested in citing these kind of tweets are mainly located close to the city in which their favorite soccer team plays.

Use-case 2: Temporal propagation analysis of tweet content. This application scenario drives the analyst to retrieve and select most relevant rules involving temporal recurrences in tweet content posting. The TweCoM framework steps of usage are similar to the ones adopted in the previous use-case. A temporal propagation analysis of tweets whose content

includes keyword *Obama* has been performed by enforcing the constraint $(Keyword, Obama) \rightarrow \{(Date, *), (Time, *)\}$ on the set of mined rules. The selected rules, among which the two reported below have been selected, highlight a typical propagation trend.

Propagation Level = 1

(vi) (Keyword, Obama) → {(Date, November 2010), (Time, from 5 p.m. to 8 p.m.} ($sup = 0.9\%$, $conf = 83\%$)

Propagation Level = 3

(vii) (Keyword, Obama) → {(Date, November 2010), (Time, p.m.)} ($sup = 0.7\%$, $conf = 62\%$)

When a significant amount of toptweets about *Obama* are posted in a limited time period, i.e., when a newsworthy political event happens, the answers/citations up to propagation level 3 are posted within the following 4-8 hours. This information may be deemed relevant for bandwidth shaping and service monitoring.

Use-case 3: Combined spatial and temporal propagation analysis of the tweet content. The last usage scenario introduces the analysis of the temporal propagation of tweet content, in conjunction with the spatial dimension. By following the same approach presented in the previous use-cases, the analyst enforces the constraint $\{(Keyword, Soccer), (Location, *)\} \rightarrow \{(Date, *), (Time, *)\}$ on the set of mined rules. He discovered, amongst others, the following rules:

Propagation Level = 2

(viii) {(Keyword, Soccer), (Location, Los Angeles)} → {(Date, November 2010), (Time, from 7 p.m. to 8 p.m.)} ($sup = 0.6\%$, $conf = 72\%$)

Propagation Level = 3

(ix) {(Keyword, Soccer), (Location, California)} → {(Date, October 2010), (Time, p.m.)} ($sup = 0.9\%$, $conf = 85\%$)

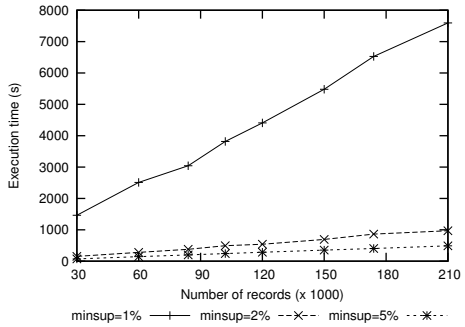


Figure 4.21: GENIO: scalability on TPC-H datasets

The above rules highlight temporal daily recurrences holding in a specific month (November 2010). Discovered rules are supported by the fact that Los Angels Galaxy team won 5 games out of 7 from October to November 2010.

To consider also spatial and temporal propagation at the same time of tweets whose content includes keyword *Obama* the analyst enforced the constraint $\{(Keyword, Obama), (Location, *) \rightarrow \{(Date, *), (Time, *)\}$ on the set of mined rules.

Propagation Level = 1

(vi) {(Keyword, Obama), (Location, Paris)} → {(Date, November 2010), (Time, p.m.)} ($sup = 0.6\%$, $conf = 62\%$)

The above rule may suggest that a number of European newsworthy events have engaged president Obama in November 2010. This information may be worth mentioning, for instance, for news recommendation.

4.5 Scalability of the GENIO algorithm

In this section the scalability of the GENIO algorithm with the number of dataset records is evaluated on synthetic datasets generated by means of the TPC-H generator [111]. By varying the scale factor parameter, tables with different cardinalities are generated. Datasets of size ranging from 30,000 to 210,000 records with 12 categorical attributes are generated The extraction of the generalized itemsets from the *lineitem* table exploits the *part*, *nation*, and *region* tables to define taxonomies on line items.

Figure 4.21, which plots the extraction time for various supports, shows that the proposed algorithm scales well also for large datasets. Since the number of extracted itemsets grows for low supports (e.g., 1%), the process becomes computationally more expensive. However, the overall CPU time is still low, less than 7600 s for the lowest considered support and largest dataset.

Chapter 5

Generalized association rule mining with constraints

This chapter addresses the enforcement of mining constraints to improve the efficiency and the effectiveness of the generalized association rule mining process. In particular, it focuses on (i) preventing the generation of redundant or less interesting patterns by pushing ad-hoc constraints into the mining process and (ii) selecting patterns valuable for analyst's purposes by means of a postprocessing step.

This chapter is organized as follows. Section 5.1 formally states the problem of generalized association rule mining with constraints and introduces two types of constraints, i.e., the schema constraints and generalized rule confidence constraints to push, respectively, into the itemset and rule mining steps. Section 5.2 presents a data mining system that is based on the discovery of generalized association rules driven by the previously introduced constraints.

5.1 Problem statement

In this section the problem of generalized association rule mining with constraints is formalized.

To further generalize the problem of extracting itemsets and rules in the presence of taxonomies, presented in Chapter 3, the concept of multiple-taxonomy is first introduced to allow the use of different generalization hierarchies over the same attribute.

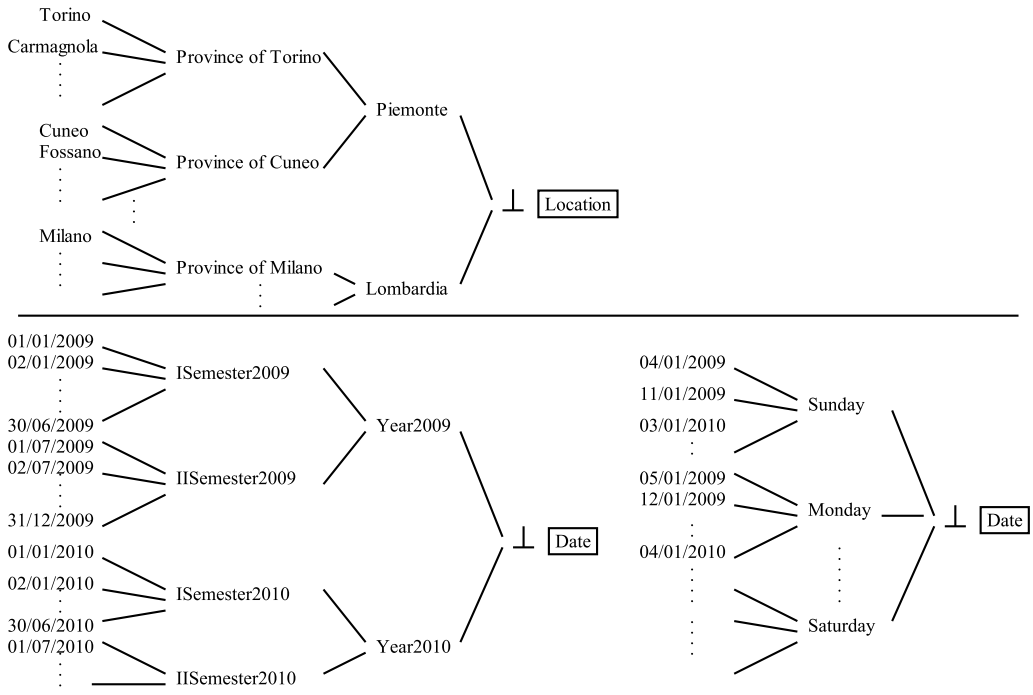


Figure 5.1: An example of multiple-taxonomy built over the example dataset

Definition 19 ***Multiple-taxonomy.*** *Let $\mathcal{T}$=$\{t_1, \ldots, t_n\}$ be a set of attributes. A multiple-taxonomy $\Theta = \{\bigcup_k AT_{1k}, \ldots, \bigcup_j AT_{nj}\}$ is a forest of aggregation hierarchies, where $\bigcup_j AT_{ij}$ is the set of generalization hierarchies defined on attribute t_i.*

The taxonomy reported in Figure 5.1 is an example of multiple-taxonomy as the date attribute is associated with two different generalization hierarchies.

The multiple-taxonomy is exploited to aggregate items during the itemset mining process.

Definition 20 ***Constraint.*** *A mining constraint is a predicate that states the characteristics of the patterns (e.g., itemsets or rules) to select during the mining process.*

For instance, the mining support and confidence thresholds are examples of constraints commonly used to reduce the amount of extracted patterns and to make the extraction and selection tasks manageable.

The generalized itemset mining process with constraints is the task of extracting all itemsets that satisfy a given set of mining constraints. Similarly, the generalized association rule mining process with constraints is the task of extracting all association rules that satisfy a given set of mining constraints.

In the following, two types of constraints are presented: (i) the schema constraints and (ii) the generalized rule confidence constraint. Their formal definitions follow.

5.1.1 The schema constraint

A schema constraint restricts the set of attributes that may appear in an itemset.

Definition 21 ***Schema constraint.*** *Let $\mathcal{T}=\{t_1, \ldots, t_n\}$ be a set of attributes. A schema constraint $Sc \subseteq \mathcal{T}$ of length k is a set of k distinct attributes. A generalized itemset X satisfies constraint Sc iff $attr(X) \subseteq Sc$, where $attr(X)$ is the set of attributes in X.*

An example of a schema constraint coming from the context-aware application domain is {user, service}. The example schema constraint specifies that the only itemsets of interest are those of length 2 of type {(user, value$_{user}$), (service, value$_{service}$)} or those of length 1 whose schema includes one of the two attributes of interest ({(user, value$_{user}$)} and {(service, value$_{service}$)}).

Definition 22 ***Schema constraint satisfaction.*** *Let X be a generalized itemset, $attr(X)$ the set of attributes in X, and $\mathcal{S}=\{Sc_1, \ldots, Sc_n\} \neq \emptyset$ a set of schema constraints. X satisfies constraints in $\mathcal{S}$ iff $attr(X) \subseteq Sc_i$ for at least an $i \in [1, n]$.*

When $\mathcal{S} = \emptyset$ no schema constraint is enforced. From the above definition, it trivially follows that if itemset X satisfies a constraint Sc, then any itemset $Y \subseteq X$ also satisfies Sc. This property can be profitably exploited to apply schema constraints during the itemset mining step.

For example the schema constraint {user, service} is satisfied by the itemset {(user,Paolo), (service,SMS)} and also by the itemsets {(user,Paolo)} and {(service,SMS)}.

5.1.2 The generalized rule confidence constraint

The generalized rule confidence constraint is exploited to prune a set of generalized rules that are considered useless. The generalized rule confidence constraint is based on the confidence measure and compares each generalized rule r with a subset $R(r)_{desc}$ of its descendants. $R(r)_{desc}$ includes all the descendant rules of r such that the body is the same body of r while the head is a descendant of the head of r. Given a minimum confidence threshold value, a generalized rule r is deemed useful if it satisfies the enforced minimum confidence threshold while none of the rules in $R(r)_{desc}$ satisfy it.

Definition 23 ***Generalized rule confidence constraint.*** *Given a minimum confidence threshold $minconf$, an arbitrary generalized rule $r : X \Rightarrow Y$ satisfies the generalized rule confidence constraint iff (i) $conf(r) \geq minconf$ and (ii) $\nexists\ r_d : X \Rightarrow Z$, with $Z \in Desc[Y]$, such that $conf(r_d) \geq minconf$.*

Given a structured dataset D, a multiple-taxonomy Θ, a set of schema constraints $\mathcal{S}$, a minimum support threshold min_sup, a minimum confidence threshold min_conf, and the corresponding generalized rule confidence constraint, the generalized association rule mining with constraints addresses the discovery of all generalized association rules satisfying all the enforced constraints.

In the following section, an example of data mining application based on the generalized rule mining process with schema and confidence constraints is presented.

5.2 The CoGAR Framework

The CoGAR (Constrained Generalized Association Rules) framework mines generalized rules satisfying a user-provided set of schema constraints and the generalized rule confidence constraint. It has been exploited to extract valuable knowledge in different application domains (e.g., network traffic analysis, mobile service analysis).

Figure 5.2 shows the building blocks of the CoGAR framework, which mainly performs three activities. In the following each activity is briefly described. (i) *Data integration and pre-processing.* Data to be analyzed is usually provided by different and heterogeneous sources. Before performing the knowledge discovery process, data is cleaned by removing irrelevant

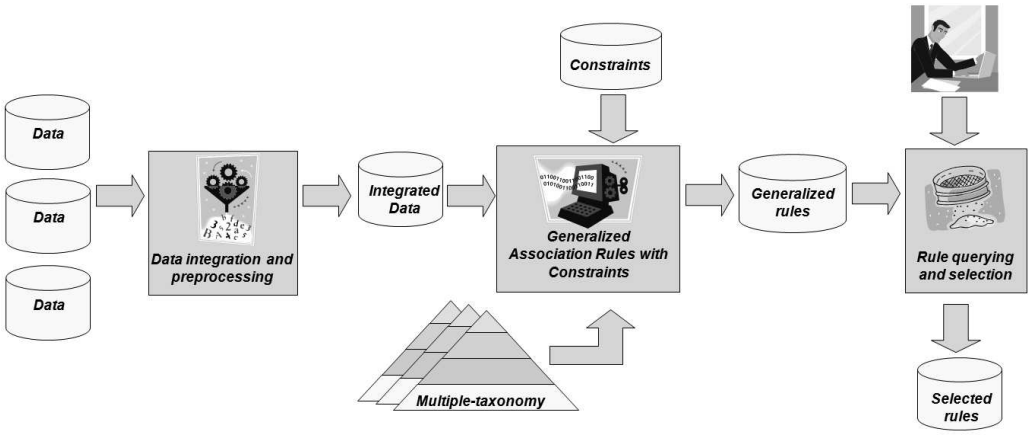


Figure 5.2: The CoGAR Framework Architecture

and redundant information and integrated into a common data structure. (ii) *Generalized association rule mining with constraints.* The mining block is the core of CoGAR. It exploits a (user-provided) multiple-taxonomy to drive the mining of generalized association rules. To tailor the mining process to specific user targets, a set of schema constraints is enforced during the itemset mining step. Moreover, the generalized rule confidence constraint is enforced to prune generalized association rules that do not provide new interesting knowledge with respect to similar rules at lower abstraction levels. (iii) *Rule querying and selection.* To allow end-users to easily exploit the mined knowledge, both the set of generalized rules and the multiple-taxonomy are stored in an XML data repository, which can be queried by means of the XQuery language [118]. A more detailed description of the functionalities of each CoGAR architectural block follows.

5.2.1 Data integration and preprocessing

This block collects data coming from different sources, integrates them, and applies preprocessing tasks (e.g., data discretization) to transform data into a common data structure. During the preprocessing phase, data cleaning and redundant data pruning may be performed. Finally, preprocessed data are stored in a common data repository. Independently of the domain, the data integration step is based on a global as view (GAV) approach. A relation global view is defined over the available sources by means of a set of semantic mappings between the schemata of the sources and the schema of the global

view. The global view provides an integrated view that is exploited by the mining block. The defined semantic mappings depend on the application domain. Hence, for each application domain, an expert of the domain must define the proper mappings, for example by means of SQL queries.

Consider a typical context-aware mobile application. In the considered application two data sources are available. The first source stores the positions of the users, while the second source stores user requests. In the first data source the position of users can be a GPS coordinate or a mobile phone cell, depending on the used device (e.g., laptops, mobile phones). A proper data transformation is needed to obtain the same value when the position of the user is semantically the same, independently of the original format (GPS coordinate or mobile phone cell). To obtain a uniform representation, each position can be, for example, mapped to the corresponding city. This semantic mapping allows obtaining a uniform representation of user locations in the global view defined over the original data sources. To obtain a global view integrating user positions and service invocations, the two data sources must be joined by exploiting the time information available in both sources. However, even in this case a proper mapping must be defined to obtain a uniform representation of the time information since the considered data sources may use different formats.

5.2.2 Constrained generalized rule miner

Given the common data repository generated by the first block of CoGAR, a multiple-taxonomy, and a set of schema constraints, this block performs the extraction of frequent generalized association rules satisfying the given schema constraints and the generalized rule confidence constraint. The extracted rules are stored in an XML repository. The mining task follows the usual two-step approach [3].

To improve the efficiency of the mining task, schema constraints should be enforced as soon as possible. An efficient itemset mining algorithm that directly mines generalized itemsets by pushing schema constraints into the itemset mining step is adopted. Generalized association rules satisfying the same schema constraints may be mined by applying any traditional rule mining algorithm on the extracted itemsets. An implementation of the traditional rule mining procedure proposed in [3] is adopted. Finally, a post-processing step is applied to enforce the generalized rule confidence constraint.

Section 5.2.4 reports detailed information about the used mining algo-

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ruleSet (rule*)>
<!ELEMENT rule (measure+, body, head)>
<!ELEMENT measure (#PCDATA)>
<!ATTLIST measure name CDATA #REQUIRED>
<!ELEMENT body (item+)>
<!ATTLIST body length CDATA #REQUIRED>
<!ELEMENT head (item+)>
<!ATTLIST head length CDATA #REQUIRED>
<!ELEMENT item EMPTY>
<!ATTLIST item attributeName CDATA #REQUIRED attributeValue CDATA #REQUIRED>
```

Figure 5.3: DTD of the XML document for association rule representation

rithms.

5.2.3 Rule querying and selection

The mining block exclusively extracts rules satisfying the enforced constraints. Further exploration may allow end-users to focus their attention on specific subsets of rules depending on their goals. The rule querying and selection block of CoGAR allows end-users to retrieve subsets of rules, or ranking them, according to their actual analysis target. XML is used to store both the extracted rules and the exploited multiple-taxonomy, while XQuery [118] is adopted to query and retrieve the rules of interest.

The DTD reported in Figure 5.3 describes the schema of the XML documents used to represent the mined rule sets. The `rule` element is used to represent rules, while the `body` and the `head` elements respectively represent the antecedent and the consequent of the rules. To store the value of the measures of interest the `measure` element is used. An XML document representing a rule set including two simple rules is reported in Figure 5.4.

The proposed XML rule representation is easily and efficiently queryable by means of the XQuery language. Figure 5.5 reports an example query, which retrieves all the rules that include the item (user, Paolo) in their body and sorts them by descending confidence. Depending on the type of rules of interest, similar queries may be easily written by the end-users of CoGAR.

Also the exploited taxonomy is represented by means of XML files. Figure 5.6 represents the DTD associated with the XML files used to store taxonomies and Figure 5.7 an XML file representing a part of a taxonomy. For each generalized item the `listOfChildren` element is used to represent its children. Each child item can be either a generalized or a not generalized item.

```
<ruleSet>
   <rule>
      <measure name="support">5.8</measure> <measure name="confidence">50.5</measure>
      <body length="1">
         <item attributeName="user">Paolo</item>
      </body>
      <head length="2">
         <item attributeName="date">2008-12-24</item>
         <item attributeName="hour">13:24</item>
      </head>
   </rule>
   <rule>
      <measure name="support">7.5</measure> <measure name="confidence">30.0</measure>
      <body length="2">
         <item attributeName="user">Tania</item>
         <item attributeName="date">2008-11-02</item>
      </body>
      <head length="1">
         <item attributeName="service">Chat</attribute>
      </head>
   </rule>
</ruleSet>
```

Figure 5.4: An example XML document representing a rule set including two rules

```
<resultSet> { for $rule in doc("MinedRuleSet.xml")/ruleSet/rule
              where exists($rule/body/item[@attributeName="user" and @attributeValue="Paolo"])
              order by $rule/measure[@name="confidence"] descending
              return $rule) } </resultSet>
```

Figure 5.5: Selection of the rules including the item (user, Paolo) in the rule body in order of decreasing confidence

Figure 5.7 reports part of a taxonomy composed of two generalization hierarchies. The first part of Figure 5.7 reports an generalization hierarchy defined on the date attribute (dates are aggregated in months, semesters, and years) while the second part defines aggregations over the service attribute.

The XML representation of the used taxonomy allows performing more complex queries by considering contemporaneously the XML file representing the mined rules and the one representing the taxonomy. For example, by using the two XML files, it is possible to select all the rules containing in

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT taxonomy (item+)>
<!ELEMENT item (listOfChildren*)>
<!ATTLIST item attributeName CDATA #REQUIRED attributeValue CDATA #REQUIRED>
<!ELEMENT listOfChildren (item+)>
```

Figure 5.6: DTD of the XML document for taxonomy representation

```
<taxonomy>
   <item attributeName="date" attributeValue="Year2010">
      <listOfChildren>
         <item attributeName="date" attributeValue="FirstSemester2010">
            <listOfChildren>
               <item attributeName="date" attributeValue="January2010"/>
               <item attributeName="date" attributeValue="February2010"/>
               .....
            </listOfChildren>
         </item>
         <item attributeName="date" attributeValue="SecondSemester2010">
            <listOfChildren>
               <item attributeName="date" attributeValue="July2010"/>
               .....
            </listOfChildren>
         </item>
      <listOfChildren>
   </item> .....
   <item attributeName="service" attributeValue="Communication">
      <listOfChildren>
         <item attributeName="service" attributeValue="PhoneCall"/>
         <item attributeName="service" attributeValue="SMS"/>
      <listOfChildren>
   </item> .....
</taxonomy>
```

Figure 5.7: An example XML document representing a taxonomy

the consequent of the rule a descendant of the generalized item (service, Communication) (see Figure 5.8).

5.2.4 Mining algorithms

The mining block of CoGAR is based on three components: (i) a schema constrained itemset mining algorithm (CI-Miner) that integrates schema constraints into the GenIO algorithm, (ii) a rule mining algorithm, and (iii) a post-processing filtering algorithm (CR-Filter). These three components are sequentially invoked.

```
<resultSet> {
  for $rule in doc("MinedRuleSet.xml")/ruleSet/rule
  where $rule/head/@length="1"
  and exists(for $item in doc("Taxonomy.xml")//item[@attributeName="service" and
             @attributeValue="Communication"]//item
             where $item/@attributeName=$rule/head/item/@attributeName
             and $item/@attributeValue=$rule/head/item/@attributeValue
             return $item)
  return $rule } </resultSet>
```

Figure 5.8: Selection of the rules including a descendant of the item (service, Communication) in the rule head

The first applied algorithm is CI-Miner. Given a structured dataset D, a multiple-taxonomy Θ, a minimum support threshold min_sup, and a set of schema constraints $\mathcal{S}$, the CI-Miner algorithm extracts frequent generalized itemsets satisfying constraints in $\mathcal{S}$ by means of an opportunistic approach, first presented in [16]. The opportunistic approach generalizes an itemset only if it is infrequent with respect to the minimum support threshold. It exploits a lazy taxonomy evaluation, which is triggered by infrequent itemsets only. Hence, an arbitrary frequent generalized itemset Y is extracted by CI-Miner iff (i) there exists at least a constraint $Sc \in \mathcal{S}$ such that $attr(Y) \subseteq Sc$ and (ii) Y has at least an infrequent descendant.

CI-Miner iteratively generates generalized itemsets by means of a level-wise approach (i.e., it is an Apriori-like approach). In an arbitrary iteration k, CI-Miner performs two steps: (i) support counting and selection of frequent itemsets with length equal to k and (ii) generation of candidate itemsets of length $k+1$ by joining k-itemsets. However, differently from traditional algorithms, at each iteration CI-Miner enforces the set of schema constraints $\mathcal{S}$ to prune the set of candidate itemsets. Only the candidate itemsets satisfying at least one of the enforced schema constraints are considered in the following iterations. This allows pruning useless candidates. CI-Miner also exploits the maximum schema constraint length to break earlier the Apriori-like mining process. In particular, itemsets longer than the maximum enforced constraint length are not extracted since they will not satisfy any constraint in $\mathcal{S}$.

The set of generalized itemsets mined by CI-Miner is exploited by our implementation of the traditional rule mining procedure proposed in [3]. Given the set of frequent generalized itemsets and a minimum confidence threshold (min_conf), the rule mining procedure generates the set of generalized association rules with a confidence at least equal to min_conf. The set of mined generalized rules is denoted as R in the following of this section.

Finally, the generalized rule confidence constraint is enforced on R by a post-processing algorithm, called CR-Filter. Algorithm 3 reports the pseudo-code of CR-Filter. To check if an arbitrary rule r satisfies the generalized rule confidence constraint, r must be exclusively compared with the set of rules with the same antecedent of r. This property is exploited by CR-Filter to enforce the generalized rule confidence constraint. Hence, rules are initially partitioned by considering their antecedent (lines 1-3). Then, the algorithm considers one rule set R_X at a time (lines 4-9) and checks which rules in R_X satisfy the generalized rule confidence constraint (lines 5-7). The initial partitioning of the rule set significantly reduces the number of comparisons.

Algorithm 3 CR-Filter: Constrained Rule Filtering

Input: set of rules R,multiple-taxonomy Θ
Output: $R_{selected}$, set of generalized rules satisfying the generalized rule confidence constraint

/*Rule partitioning by considering their antecedent*/
1: **for all** itemset X in $\{X|r : X \Rightarrow Y \in R\}$ **do**
2: $R_X = \{r|r \in R$ and $r.antecedent = X\}$ /*R_X is the set of rules with the itemset X as antecedent*/
3: **end for**

/*Enforcement of the generalized rule confidence constraint*/
4: **for all** rule set R_X **do**
5: **for all** rule r in R_X **do**
6: delete r from R_X if $\exists\ r_l \in R_X$ such that $r_l.consequent$ is a descendant of $r.consequent$
7: **end for**
8: $R_{selected}$=$R_{selected} \cup R_X$
9: **end for**
10: **return** $R_{selected}$

5.3 CoGAR application to context-aware data

Evaluation experiments of the CoGAR framework in the context-aware domain were performed both on two datasets obtained from a real context-aware application for mobile service provisioning domain. In Section 5.3.1 the main characteristics of these datasets and the set of enforced schema constraints are reported, while, in Section 5.3.2, the effectiveness of the presented system in supporting knowledge discovery from context-aware data has been validated by a domain expert.

All the experiments were performed on a 2.66 GHz Pentium IV system with 8 GB RAM, running Ubuntu Release 9.10. The CoGAR framework was implemented in the Python programming language [95].

5.3.1 Context-aware data and constraints

Telecom Italia Lab[1] provided us the *Recs* dataset containing a set of requests submitted to a real context-aware service provider system (the *Recs* system). The *Recs* system provides recommendations to mobile device users on restaurants, museums, movies, and other entertainment activities. Each user can request a recommendation (GET_REC service), enter a score (VOTE service), or update a score (UPDATE_VOTE service) for an entertainment activity. The analyzed dataset was obtained by logging the requests of 20 users and their locations over a time period of three months. The dataset contains 5814 records (i.e., requests). Each record is characterized by the request type, the parameters of the request, the user and its context information

[1]TILab is the Telecom Italia Group research hub

Table 5.1: CoGAR: example schema constraints on the Recs dataset

Constraint	Rule Example
$\{user, date, time\}$	$\{(user, John)\} \Rightarrow \{(date, June), (time, morning)\}$
$\{user, time, place\}$	$\{(user, John)\} \Rightarrow \{(time, morning), (place, office)\}$
$\{user, time, param\}$	$\{(user, John), (time, morning)\} \Rightarrow \{(param, OUT)\}$
$\{user, date, param\}$	$\{(user, John), (date, winter)\} \Rightarrow \{(param, OUT)\}$
$\{user, date, place\}$	$\{(user, John)\} \Rightarrow \{(date, winter), (place, office)\}$
$\{user, place, param\}$	$\{(user, John), (place, office)\} \Rightarrow \{(param, OUT)\}$
$\{user, service, time\}$	$\{(user, John), (service, CALL)\} \Rightarrow \{(time, 2-6p.m.)\}$
$\{user, service, date\}$	$\{(user, John), (service, CALL)\} \Rightarrow \{(date, December)\}$
$\{user, service, place\}$	$\{(user, John), (service, CALL)\} \Rightarrow \{(place, office)\}$
$\{user, service, param\}$	$\{(user, John), (service, CALL)\} \Rightarrow \{(param, OUT)\}$
$\{service, date, time\}$	$\{(service, CALL), (date, winter), (time, afternoon)\}$
$\{service, place, time\}$	$\{(service, WEATHER)\} \Rightarrow \{(place, home), (time, evening)\}$
$\{service, place, date\}$	$\{(service, WEATHER), (place, home)\} \Rightarrow \{(date, summer)\}$
$\{service, param, date\}$	$\{(service, CALL)\} \Rightarrow \{(param, OUT), (date, week-end)\}$
$\{service, param, time\}$	$\{(service, CALL)\} \Rightarrow \{(param, OUT), (time, afternoon)\}$
$\{service, place, param\}$	$\{(service, WEATHER), (place, home)\} \Rightarrow \{(param, TODAY)\}$

(user location, date, and time). To perform generalized rules mining, a single taxonomy including the following generalization hierarchies has been initially defined. The usage of a multiple-taxonomy is discussed in Section 5.3.2.

- date → month → trimester → year
- time stamp → hour → timeslot (two hours timeslots)→ day period (AM/PM)
- latitude:longitude → city → country

Enforced schema constraints A domain expert in charge of service provisioning provided us the schema constraints reported in Table 5.1, together with examples of compliant rules. He was very interested in profiling users and services. Thus, constraints include the user and/or the service attribute. The user attribute is exploited to characterize the user behavior, while the service attribute is exploited to profile service usage. Additional information deemed relevant was the user/service context information (e.g., service invocation date and time, user location).

5.3.2 Knowledge discovery from context-aware data

The habits of specific users (or user categories) may be characterized by some kind of recurrence. By selecting from Table 5.1 only the schema constraints involving the *user* attribute (i.e., the first ten schema constraints), generalized rule mining is tailored to user profiling. For example, the following

generalized rule allows the discovery of valuable knowledge about a generic user of the *Recs* application, named *Rossi*[2]. It has been mined by enforcing a support threshold equal to 1% (i.e., absolute threshold=58) and by exploiting the user-provided taxonomy proposed in Section 5.3.1.

{ `(user, Rossi)`} ⇒ { `(date, II Trimester 2009) (service, GET_REC)` } ($sup = 9.8\%, conf = 95\%$)

This rule highlights that user *Rossi* is interested in getting recommendations in a specific time period, with rule confidence 95%. Thus, it provides relevant knowledge on this user attitudes. In particular, for specific users (user categories), rules satisfying the above constraints may highlight the service type users are mainly interested in, the context in which requests are commonly submitted, and the parameters which are frequently used.

Consider now the following rule:

{ `(user, Rossi)`} ⇒ { `(date, Year 2009) (service, GET_REC)` } ($sup = 10.4\%, conf = 98\%$)

It highlights a higher level recurrence that does not provide additional knowledge with respect to the former one, as the confidence increase from the former to the latter one is exclusively due to the rule head generalization on the date attribute (i.e., from trimester to year). The generalized rule confidence constraint enforcement allows sharply pruning this kind of rules that may be considered redundant and, thus, not useful for analyst decision making.

By enforcing all the constraints reported in Table 5.1, the following generalized rule may also be extracted.

{ `(location, Italy) (date, II Trimester 2009)`} ⇒ {`(service, GET_REC)`} ($sup = 17\%, conf = 97\%$)

This rule shows a different application-oriented recurrence. In particular, it emphasizes that the *GET_REC* service is frequently requested when the location is Italy and the date is in the second semester of year 2009.

[2]Actual individual names are not provided for privacy reasons.

Multiple-taxonomy evaluation to enhance the knowledge discovery

For several application domains the investigation on different facets of the same feature may be desirable. Multiple-taxonomy evaluation provides the capability to explore at the same time different generalization hierarchies on the same attribute. For example, suppose to enrich the taxonomy proposed in Section 5.3.1 by extending the *date* attribute taxonomy with the week day, week, and two-month time period information as follows.

- date → month → trimester → year
- date → two-month time period
- date → week day → week
- time stamp → hour → timeslot (two hours timeslots)→ day period (AM/PM)
- latitude:longitude → city → country

The usage of different generalization hierarchies for the *date* attribute could be semantically interpreted as a faceted knowledge classification, in which facets are different axes along which items are aggregated.

By enforcing a minimum support threshold equal to 2% and the same constraints proposed in Section 5.3.1, the following generalized association rules are mined (among others).

{ `(user, Rossi)` } ⇒ { `(date, Tuesday)`, `(service, GET_REC)` } ($sup = 3.4\%, conf = 33\%$)
{ `(user, Rossi)` } ⇒ { `(date, May-June 2009)`, `(service, GET_REC)` } ($sup = 7.5\%, conf = 73\%$)

These rules provide a more detailed knowledge on user *Rossi* habits. They emphasize a different facet of the *date* attribute (i.e., the day of the week) that may better support domain experts in user profiling (e.g., to personalize daily promotions depending upon the yearly time period). They also allow the specialization of previously mined knowledge by highlighting a smaller time slice (i.e., May-June vs. May-July). When a single generalization hierarchy per attribute is allowed, multiple extractions are needed to obtain the same information.

5.4 CoGAR application to network traffic data

Evaluation experiments of the CoGAR framework in the context of network traffic analysis were performed both on real network traffic captures. In Section 5.4.1 the main characteristics of these datasets and the set of enforced schema constraints are reported, while, in Section 5.4.2, the effectiveness of the presented system in supporting knowledge discovery from network traffic data has been validated by a domain expert.

All the experiments were performed on a 2.66 GHz Pentium IV system with 8 GB RAM, running Ubuntu Release 9.10. The CoGAR framework was implemented in the Python programming language [95].

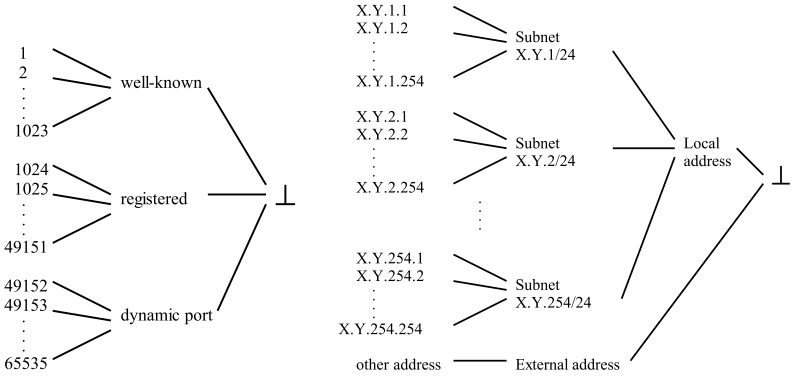
5.4.1 Network traffic data and constraints

NetCapture is a network traffic dataset obtained by performing different capture sessions with the open-source Network Analyzer tool [87] on a backbone link of a campus network. Captured traffic has been aggregated in traffic flows (i.e., records summarizing a group of similar and temporally contiguous packets). Each flow is characterized by six attributes: Source IP address, destination IP address, source port, destination port, flow size (i.e., the size of the flow expressed in byte), and number of IP packets aggregated in that flow. The *NetCapture* dataset is characterized by 16,783 records.

The taxonomy used in the experiments aggregates infrequent items according to the following generalization hierarchies. (1) Source and destination ports are aggregated by exploiting the generalization hierarchy shown in Figure 5.9(a), which introduces three aggregation values (i.e., *well known, registered, dynamic*). (2) Source and destination IP addresses are aggregated by exploiting the aggregation tree shown in Figure 4.1[3]. IP addresses are aggregated in *subnet* if they are local to the campus network. IP addresses not belonging to the campus network are aggregated in a more general *external address* node. Furthermore, both the flow size (bytes) and the number of IP packets attributes are uniformly discretized in 4 bins, whose intervals are [1,1000), [1000, 2000), [2000, 3000), and equal or greater than 3000.

Enforced schema constraints A network analyst was interested in identifying pairs of network devices (each device is identified by its IP) that frequently communicated together and the characteristics of the generated

[3]For privacy reasons, the first 16 bits of the IP addresses are hidden.



(a) Generalization hierarchy AT_{port} for the source and destination port attributes

(b) Generalization hierarchy $AT_{IP-address}$ for the source and destination IP address attributes

Figure 5.9: Generalization hierarchies for the port and IP attributes

Table 5.2: CoGAR: schema constraints on the *NetCapture* dataset

Constraint	**Rule example**
$\{IPsource, IPdest, Portsource\}$	$\{(IPsource, X.Y/16), (Portsource, 184)\} \Rightarrow \{(IPdest, Extern)\}$
$\{IPsource, IPdest, Portdest\}$	$\{(IPsource, X.Y/16)\} \Rightarrow \{(IPdest, Extern), (Portdest, 184)\}$
$\{IPsource, Portsource, Flowsize\}$	$\{(IPsource, X.Y/16), (Portsource, 50)\} \Rightarrow \{(Flowsize, 10)\}$
$\{IPdest, Portdest, Flowsize\}$	$\{(IPdest, X.Y/16), (Portdest, 50)\} \Rightarrow \{(Flowsize, 10)\}$
$\{IPsource, Portsource, PacketsNr\}$	$\{IPsource, X.Y/16), (Portsource, 50)\} \Rightarrow \{(PacketsNr, 10)\}$
$\{IPdest, Portdest, PacketsNr\}$	$\{(IPdest, X.Y/16), (Portdest, 50)\} \Rightarrow \{(PacketsNr, 10)\}$
$\{IPsource, IPdest, PacketsNr\}$	$\{(IPsource, X.Y/16), (IPdest, Extern)\} \Rightarrow \{(PacketsNr, 10)\}$
$\{IPsource, IPdest, FlowSize\}$	$\{(IPsource, X.Y/16), (IPdest, Extern)\} \Rightarrow \{(Flowsize, 10)\}$
$\{Portsource, Portdest, FlowSize\}$	$\{(Portsource, 1024), (Portdest, 184)\} \Rightarrow \{(Flowsize, 10)\}$
$\{Portsource, Portdest, PacketsNr\}$	$\{(Portsource, 1024), (Portdest, 184)\} \Rightarrow \{(PacketsNr, 10)\}$

traffic (e.g., used ports, number of transmitted packets). He was also interested in identifying single network devices (IPs) with a huge amount of incoming/outgoing data on specific ports. He provided us the schema constraints reported in Table 5.2.

5.4.2 Knowledge discovery from network traffic traces

Consider a network traffic capture performed by an analyst for monitoring incoming traffic flows by means of generalized association rule extraction. To discover interesting correlations hidden in the network traffic trace, the itemset mining process might be initially driven by the whole schema constraint set reported in Table 5.2 and by a minimum support threshold equal

to min_sup=1.5%, while the rule generation should be driven by a minimum confidence threshold min_conf=10% and the generalized rule confidence constraint. Among others, the following generalized rule is extracted.

(i) { `(IP source, Extern), (Flow Size,` > 3000`)` } $\Rightarrow$ { `(IP destination, X.Y.85/24)` } ($sup = 1.7\%, conf = 15\%$)

The above rule highlights significant incoming external traffic flows to subnet $X.Y.85/24$. Indeed, the network domain expert should consider to monitor the most significant incoming flows involving subnet $X.Y.85/24$ to understand problems coming from network traffic overloading.

Beyond the former rule, a traditional generalized rule miner, driven by support and confidence constraints only, would extract the following higher level rule as well:

(ii) { `(IP source, Extern), (Flow Size,` > 3000`)` } $\Rightarrow$ { `(IP destination, X.Y/16)` } ($sup = 2.8\%, conf = 32\%$)

Its extraction may mislead the expert in decision making, as it could lead him to carry out a monitoring campaign on a larger set of IP addresses ($X.Y/16$) even if the contemporaneous extraction of (i) suggests to restrict the monitoring space to the 24-bit subnet $X.Y.85/24$. The enforcement of the generalized rule confidence allows avoiding redundant and possibly misleading knowledge extraction, thus, easing the knowledge discovery process.

A more insightful analysis tailored to traffic volume monitoring may focus on how hosts belonging to subnet $X.Y.85/24$ are contacted on specific well-known ports (e.g., port 1000). By focusing on recurrences involving couples source/destination IP addresses and ports only (i.e., enforcing just the first two mining constraints in Table 5.2) and by lowering the support thresholds (min_sup=0.1%), the following rule is mined.

(iii) { `(IP destination, X.Y.85.189)` } $\Rightarrow$ { `(Destination Port, 1000)` } ($sup = 1.2\%, conf = 88.8\%$)

It highlights relevant incoming connections through well-known port 1000 of internal IP addresses both belonging to subnet $X.Y.85/24$. The analyst may deem this knowledge relevant as he monitors service usage on specific ports to prevent and manage network overloading situations.

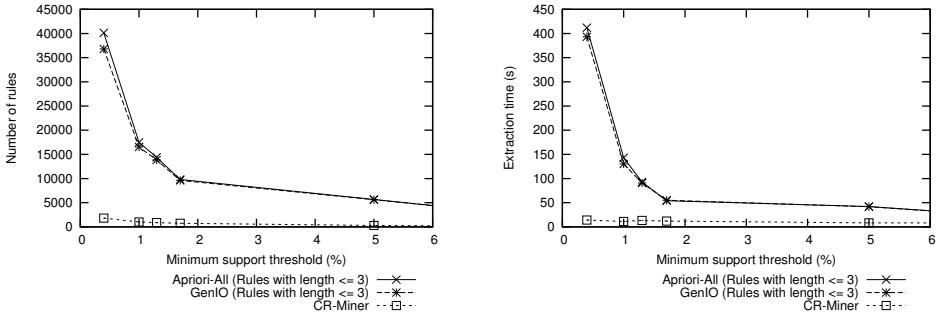
5.5 CoGAR performance analysis

The value of the minimum support and confidence thresholds significantly affect the number of extracted itemsets and, consequently, the number of mined rules. To avoid the extraction of uninteresting correlations, two types of constraints, i.e., the analyst-provided schema constraints and the generalized rule confidence constraint, have been introduced. In this section, the effect of the enforced constraints in terms of the number of extracted generalized rules is discussed separately (see Sections 5.5.1 and 5.5.2). Furthermore, the scalability of the CI-Miner algorithm on synthetic data is also analyzed (see Section 5.5.3).

5.5.1 Effect of the schema constraints

The CoGAR framework exploits the CI-Miner algorithm to perform generalized itemset mining. Then, as discussed in Section 4.1, a traditional rule mining procedure is used to generate the rule set. Itemset mining is commonly constrained by a minimum support constraint. The CI-Miner algorithm also enforces a schema constraint to further reduce the amount of uninteresting extracted itemsets and, consequently, the number of rules. Figures 5.10(a) and 5.11(a) report for the *Recs* and *Netcapture* datasets (i) the number of mined rules and (ii) the corresponding extraction time when varying the minimum support threshold and enforcing no confidence threshold (i.e., min_conf=0). We compared the number of rules mined by the mining block based on CI-Miner (i.e., the rules satisfying the enforced constraints and the minimum thresholds) with the number of rules mined by exploiting both *Apriori-All* [103] and GenIO [16] in the initial itemset set mining phase. Unlike CI-Miner, *Apriori-All* and GenIO do not enforce any schema constraints, indeed a post-processing step is required to extract the same knowledge of interest. To perform a fair comparison between the three extraction processes, we limited the maximum length of the mined itemsets (and rules) to the maximum constraint length (i.e., max_len=3 for schema constraints in Tables 5.1 and 5.2) also when the *Apriori-All* and GenIO algorithms are executed. The enforcement of schema constraints into the mining process significantly reduces the amount of extracted irrelevant knowledge, thus improving the efficiency of the knowledge discovery process.

GenIO slightly outperforms *Apriori-All*, at small and medium support thresholds (e.g., min_sup=1%), in terms rule pruning selectivity due to the support-driven approach to pattern generalization [16]. For the *Recs* dataset



(a) Number of extracted rules

(b) Extraction time

Figure 5.10: CoGAR: effect of the minimum support threshold on the number of mined rules and corresponding execution time. *Recs* dataset

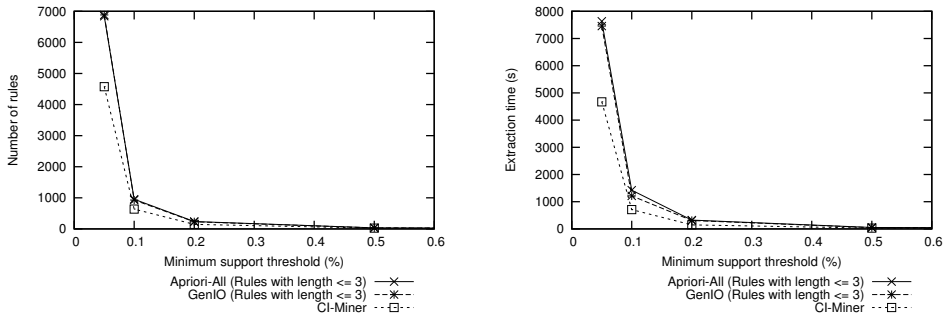
(see Figures 5.10(a)) and 5.10(b)), schema constraint enforcement yields a reduction larger than 90% of the rule cardinality for low support thresholds (e.g., *min_sup*=0.3%) with respect to both *Apriori-All* and GenIO. The corresponding time reduction is significant also for medium support thresholds, while it becomes more and more relevant when further decreasing the minimum support threshold.

Similar considerations hold for the *NetCapture* dataset (see Figure 5.11(a)) and 5.11(b)). Mined rule set cardinality reduction and time reduction are less significant for the *NetCapture* dataset, with respect to the ones obtained on *Recs*, because *NetCapture* is characterized by fewer attributes than *Recs*. Thus, the number of extracted rules is lower and the corresponding time reduction is less significant.

5.5.2 Effect of the generalized rule confidence constraint

To perform rule generation from the set of extracted frequent itemsets, the CoGAR framework exploits an our efficient implementation of the traditional rule mining procedure proposed by [3] followed by the CR-Filter post-pruning algorithm. The traditional rule mining step [3] is constrained by a minimum confidence threshold. Besides, we enforced the generalized rule confidence constraint to further prune the set of extracted generalized rules.

Figure 5.12(a) reports the impact of the support and confidence thresholds on the number of rules mined from the *Recs* dataset when the generalized



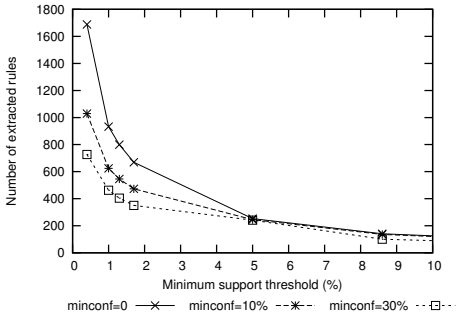
(a) Number of extracted rules

(b) Extraction time

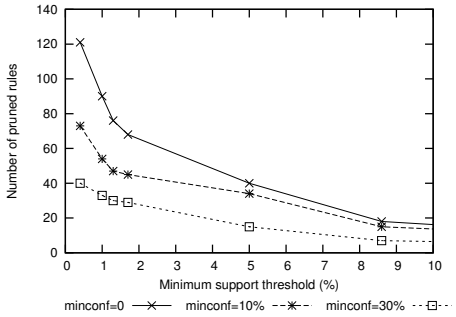
Figure 5.11: CoGAR: effect of the minimum support threshold on the number of mined rules and corresponding execution time. *NetCapture* dataset

rule confidence constraint is enforced.

As expected, the pruning selectivity is more relevant when higher support and confidence thresholds are enforced. The generalized rule confidence constraint enforcement focuses on further prune a subset of the mined rule set that (i) includes at least a generalized item in the rule consequent, and (ii) may be considered as redundant (Cf. Definition 23). To evaluate the pruning selectivity of the new constraint, Figure 5.12(b) reports, for the *Recs* dataset, the number of rules pruned by enforcing the generalized rule confidence constraint and by varying the support and confidence thresholds. A significant pruning effectiveness (i.e., a pruning rate in the range [6%-12%] for every combination of support and confidence values) clearly comes out when a high number of (possibly redundant) generalized rules is extracted. This is the case of the *Recs* dataset, for which around 65%-80% of the extracted rules (with any support and confidence thresholds) contain at least a generalized item in the rule head. On the contrary, for the *Netcapture* dataset, such a percentage is rather limited (e.g., even less than 10% for some support and confidence combinations). Indeed, the generalized rule confidence constraint enforcement becomes no more interesting and, thus, the corresponding graphs have been omitted. The balancing between the confidence threshold and the new generalized rule confidence constraint could be highlighted by comparing the curves reported in Figure 5.12(b). When no minimum confidence is enforced (i.e., $minconf$=0), the generation of a (lower level) rule, satisfying the minimum support threshold, prevents the generation of all the frequent rules characterized by (i) the same body, and (ii) an ancestor (i.e., higher level itemset) of its rule head. Indeed, the generalized rule confidence



(a) *Recs* dataset. Generated rules with the generalized rule confidence constraint



(b) *Recs* dataset. Number of pruned rules with the generalized rule confidence constraint

Figure 5.12: CoGAR: effect of the generalized rule confidence constraint. *Recs* dataset

constraint pruning effectiveness is maximum. When, instead, the confidence threshold increases, some of the lower level rules are discarded due to the confidence constraint and, thus, the pruning effectiveness of the generalized rule confidence constraint decreases.

We also analyzed the execution time of the two steps of the mining activity separately (i.e., itemset mining constrained by the minimum support threshold and schema constraints and rule mining constrained by both confidence threshold and the new generalized rule confidence constraint). On average, the execution time of the itemset mining step typically accounts for more than 90% of the total execution time, while the remaining time is devoted to the rule mining and post-processing step. Indeed, the rule extraction process, also including the post-processing phase needed for generalized rule confidence constraint enforcement, still remains the less computationally intensive mining step.

5.5.3 Scalability of the mining process

This section analyzes the scalability of the rule mining process in terms of (i) the number of dataset records, and (ii) the taxonomy height. To perform the scalability analysis a synthetic data generator based on the IBM data generator [63] is exploited.

To allow generating taxonomies of different heights, the original code of

the synthetic generator is properly extended. The taxonomy is generated by means of the following procedure. For each attribute, all its dataset values are considered as leaves (i.e., level 1) of the corresponding generalization hierarchy. Next, attribute values are sorted in a lexicographical order and grouped together based on a constant aggregation factor f. Finally, each group of items is collapsed in a newly generated upper level item. The above procedure is iterated until a unique root is available. For all attributes, we set the factor f to $\lceil \sqrt[(h - 1)]{n} \rceil$, where n is attribute domain cardinality. This leads to the creation of a generalization hierarchy, composed of h aggregation levels, such that the ratio between the number of items at level l and the number of items at level $l - 1$ keeps constant.

Since some attributes are continuous, we performed a discretization step based on an equi-width technique by setting the number of bins to 10.

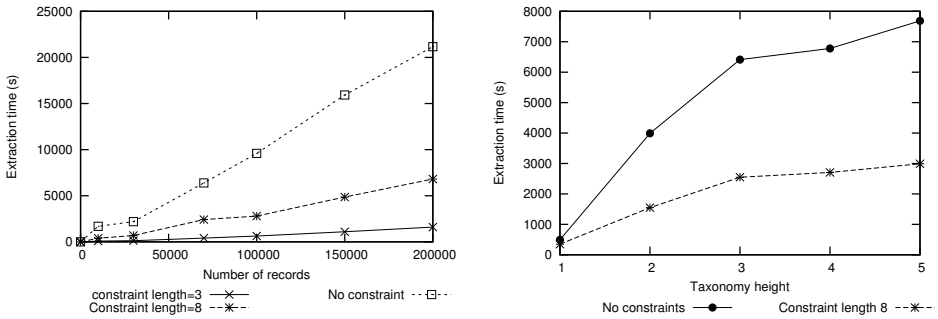
Scalability with respect to the cardinality of the dataset

To analyze the scalability of our approach with respect to the cardinality of the dataset, datasets of size ranging from 5,000 to 210,000 records with 12 categorical attributes and corresponding taxonomies having height equal to 5 are generated. A minimum support threshold equal to 1% was enforced during the itemset mining step, while the generalized rule confidence constraint and no minimum confidence threshold (i.e., min_conf=0) were enforced during the rule generation step. Three different schema constraint configurations are evaluated: (i) All the possible combinations of the first three attributes, (ii) all the possible combinations of the first eight attributes, and (iii) no schema constraints.

Figure 5.13(a) plots the extraction time (i.e., comprehensive of itemset and rule mining time and rule post-processing time) for the different constraint settings by varying the number of records. It shows that the proposed algorithm scales almost linearly with respect to the number of records. The overall CPU time is still acceptable also when dealing with larger datasets, and even when no constraint is enforced. Obviously, the number and type of schema constraints significantly impact on the execution time.

Scalability with respect to the taxonomy height

To evaluate the impact of the taxonomy height, different taxonomies with height ranging from 1 to 5 are generated. For each attribute, a 5-level generalization hierarchy is synthetically generated by means of the procedure



(a) Scalability in terms of the number of records. Taxonomy height=5.

(b) Scalability in terms of the taxonomy height. Number of records=70, 000.

Figure 5.13: CoGAR: scalability of the mining process on the IBM datasets min_sup=1%. min_conf=0

described in Section 5.5.3. Next, the top level is pruned and a 4-level generalization hierarchy is generated. By iteratively applying this procedure on each generalization hierarchy, the taxonomy height is reduced by one at each iteration. At the end of the generation process, five different taxonomies of decreasing height are available for testing.

The number of records is set to 70, 000. A minimum support threshold equal to 1% and the new generalized rule confidence constraint are enforced, respectively, during the itemset and rule mining steps, while no minimum confidence threshold is enforced. Two different schema schema constraint configurations have been evaluated: (i) All the combinations of the first eight attributes, and (ii) no schema constraint. Figure 5.13(b) plots the extraction time by varying the taxonomy height. Since each increase of the taxonomy height corresponds to an increase of the total number of items, when increasing the taxonomy height also the number of extracted rules and the execution time increase. However, the increase does not scale linearly with respect to the taxonomy height. The difference between the number of items of the l-th taxonomy level and the number of items of the (l-1)-th taxonomy level depends on the value of l. According to the taxonomy generation process, the higher is the value of l, the lower is the difference between the number of items of the two considered taxonomies. As expected, the execution time increase is higher when moving from height 1 to 2, while it is lower for higher height values.

The taxonomy height also affects the computational cost of the post-

processing steps. The time spent in rule post-processing scales roughly linearly with the taxonomy height. However, as already discussed, its impact on the overall execution time is negligible.

Chapter 6

Change mining by means of generalized patterns

This chapter addresses the task of change mining in the context of generalized itemsets. The problem of discovering relevant data recurrences and their most significant temporal trends is becoming an increasingly appealing research topic. The application of frequent itemset mining and association rule extraction algorithms [2] to discover valuable correlations among data has been thoroughly investigated in a number of different application contexts (e.g., market basket analysis [2], medical image processing [7]). In the last years, the steady growth of business-oriented applications tailored to the extracted knowledge has prompted the need of analyzing the evolution of the discovered patterns. Since, in many business environments, companies are expected to reactively suit product and service provision to customer needs, the investigation of the most notable changes between the set of frequent itemsets or association rules mined from different time periods has become an appealing research topic [6, 35, 44, 80, 98, 114].

Frequent itemset mining activity is constrained by a minimum support threshold to discover patterns whose observed frequency in the source data (i.e., the support) is equal to or exceeds a given threshold [2]. However, the enforcement of low support thresholds may entail generating a very large amount of patterns which may become hard to look into. On the other hand, higher support threshold enforcement may also prune relevant but not enough frequent recurrences. To overcome these issues, generalized itemset extraction could be exploited. Generalized itemsets, which have been first introduced in [103] in the context of market basket analysis, are itemsets that provide a high level abstraction of the mined knowledge. By exploiting

a taxonomy (i.e., a is-a hierarchy) over data items, items are aggregated into higher level (generalized) ones.

Change mining in the context of frequent itemsets may exploit generalized itemsets to represent patterns that become rare with respect to the support threshold, and thus are no longer extracted, at a certain point. Previous approaches allow both keeping track of the evolution of the most significant pattern quality indexes (e.g., [6, 21, 26]) and discovering their most fundamental changes (e.g., [10, 44, 80]). Itemset generalization may allow preventing infrequent knowledge discarding. At the same time, experts may look into the information provided by the sequence of generalizations or specializations of the same pattern in consecutive time periods.

This chapter presents a kind of dynamic pattern, namely the HiGens (History GENeralized Pattern) based on generalized patterns [31]. A HiGen compactly represents the minimum sequence of generalizations needed to keep knowledge provided by a not generalized itemset frequent, with respect to the minimum support threshold, in each time period. If an itemset is frequent in each time period, the corresponding HiGen just reports its support variations. Otherwise, when the itemset becomes infrequent at a certain point, the HiGen reports the minimum number of generalizations (and the corresponding generalized itemsets) needed to make its covered knowledge frequent at a higher level of abstraction.

In case an infrequent not generalized itemset has multiple generalizations belonging to the same minimal aggregation level, many HiGens associated with the same itemset are generated. To focus the attention of the analysts on the frequent generalizations of a rare itemset covering the same knowledge with a minimal amount of redundancy and, thus, reduce the number of the generated patterns, a more selective type of HiGen, i.e., the Non-redundant HiGen, is presented as well [31]. Non-redundant HiGens are HiGens that include, for each time period, the frequent generalizations of the reference itemset of minimal generalization level characterized by minimal support.

This chapter is organized as follows. Section 6.1 introduces preliminary notions and formally states the HiGen mining problem. Section 6.2 describes the HiGen Miner algorithm, Section 6.3 analyzes the performance of the HiGen Miner algorithm on synthetic data, Section 6.4 evaluates the efficiency and the effectiveness of the proposed approach in the context-aware domain, while Section 4.4 presents a framework that exploits HiGen mining in social network analysis.

6.1 Problem statement

This section introduces main notions concerning dynamic generalized itemset mining from structured data and formally states the HiGen mining problem.

A structured dataset is a collection of records, where each record is a set of pairs $(attribute, value)$ (e.g., *(date, 5:05 p.m.)*), called items, which identify specific data features (e.g., the time) and their values in the corresponding domains (e.g., *5:05 p.m.*). Dynamic data mining considers datasets associated with different time periods. In the following, the notion of timestamped structured dataset is introduced.

Definition 24 Timestamped structured dataset. *Let $\mathcal{T}=\{t_1, t_2, \ldots, t_n\}$ be a set of data features and $\Omega=\{\Omega_1, \Omega_2, \ldots, \Omega_n\}$ its corresponding domains. t_i may be either a categorical or a numeric discrete data feature. Let $W = [w_{start}, w_{end}]$ be a time interval. A timestamped structured dataset D is a collection of records, where (i) each record r is a set of pairs $(t_i, value_i)$ where $t_i \in \mathcal{T}$ and $value_i \in \Omega_i$, (ii) each $t_i \in \mathcal{T}$, also called attribute, may occur at most once in any record, and (iii) each record has a time stamp $r_{ts} \in W$.*

In the case of datasets with continuous attributes, the value range should be discretized into intervals, and the intervals mapped into consecutive positive integers. Consider again the example datasets D_1 and D_2 reported in Tables 6.1(a) and 6.1(b). They are examples of timestamped structured datasets composed of 4 attributes (i.e., date, time, location, and product description), among which the date attribute is selected as time stamp. Dataset D_1 includes product sales (i.e., records) whose time stamp ranges from 2009-01-01 to 2009-01-31 (see Table 6.1(a)), while D_2 includes product sales whose time stamp ranges from 2009-02-01 to 2009-02-28 (see Table 6.1(b)).

Generalized itemset mining exploits a set of generalization hierarchies, i.e., a taxonomy, built over data items to generalize at higher levels of abstraction items represented in a timestamped structured dataset. A taxonomy is a is-a hierarchy built over data item values composed of generalization hierarchies defined on each data attribute.

Taxonomies may be either analyst-provided or inferred by means of ad-hoc algorithms (e.g., [36, 49]). Figure 6.1 reports three examples of generalization hierarchies constructed over the attributes of the example timestamped datasets reported in Tables 6.1(a) and 6.1(b), exception for the time stamp attribute (i.e., the date) which, by construction, is not considered.

According to Definition 3, the set of all the generalization hierarchies in Figure 6.1 is a taxonomy.

Date	Time	Location	Product description
2009-01-02	11:00 p.m.	Turin	T-shirt
2009-01-03	11:10 p.m.	Rome	T-shirt
2009-01-05	8:40 a.m.	Paris	Jacket
2009-01-30	5:05 p.m.	Paris	Jacket
2009-01-31	8:40 a.m.	Cannes	Jacket

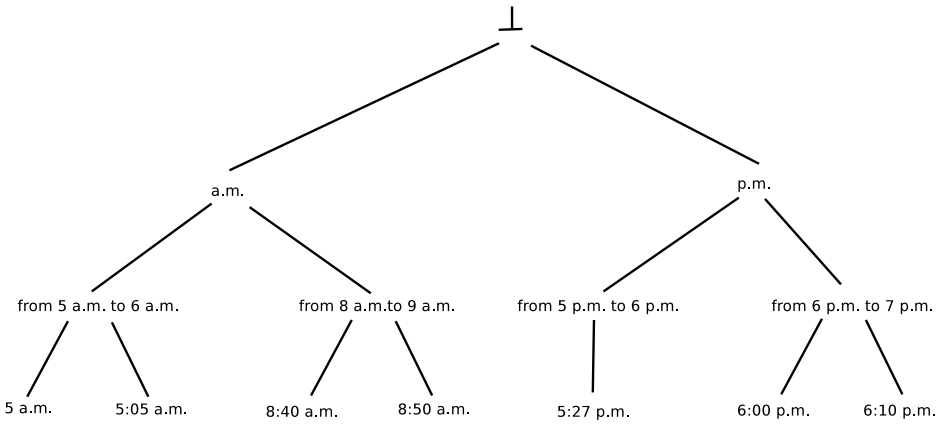
(a) Dataset D_1. Product sales in January 2009

Date	Time	Location	Product description
2009-02-02	11:10 p.m.	Turin	T-shirt
2009-02-02	10:27 p.m.	Turin	T-shirt
2009-02-05	8:50 a.m.	Paris	Jacket
2009-02-05	5:00 p.m.	Cannes	Jacket
2009-02-28	5:00 p.m.	Rome	Jacket

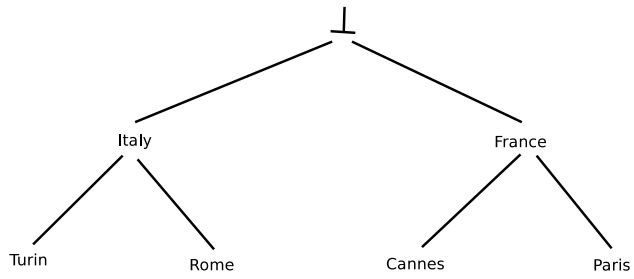
(b) Dataset D_2. Product sales in February 2009

Table 6.1: Examples of timestamped structured datasets

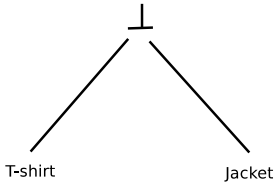
The frequent (generalized) itemset mining problem [103] focuses on discovering, from the analyzed data, generalized and not generalized itemsets (Cf. Definition 6) whose support value is equal to or exceeds a minimum support threshold. Given an ordered sequence of timestamped structured datasets (Cf. Definition 24) relative to different time periods, a taxonomy (Cf. Definition 3), and a minimum support threshold, dynamic change mining [6], in the context of frequent itemsets, investigates the changes and the evolution of the extracted itemsets, in terms of their main quality indexes (e.g., the support), from one time period to another. The dynamic change mining problem, in the context of frequent itemsets, may be extended by exploiting frequent generalized itemsets to represent knowledge associated with infrequent patterns. In the following, the concepts of HiGen and Non-redundant HiGen are introduced.



(a) Time - Generalization Hierarchy GH_{time}



(b) Location - Generalization Hierarchy $GH_{location}$



(c) Product description - Generalization Hierarchy $GH_{product}$

Figure 6.1: Taxonomy over data items in the example datasets D_1 and D_2

6.1.1 The HiGen

A HiGen, associated with both a not generalized itemset *it* and an ordered sequence of timestamped datasets $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$, is a ordered sequence of generalized itemsets $g_1, g_2, \ldots, g_n$ that represents the evolution of the knowledge covered by *it* in $\mathcal{D}$. Each g_i is a frequent (generalized) itemset extracted from D_i. g_i may be either (i) *it*, in case *it* is frequent in D_i with respect to the minimum support threshold, or (ii) one of the frequent generalizations of *it* in D_i characterized by minimum generalization level otherwise. A more formal definition of HiGen follows.

Definition 25 HiGen. *Let $\mathcal{D}$=$\{D_1, D_2, \ldots, D_n\}$ be an ordered sequence of timestamped structured datasets and Γ a taxonomy built over data items in $D_i \in \mathcal{D}$ $\forall$ i. Let it be a not generalized itemset and min_sup be a minimum support threshold. A* HiGen *HG_{it}, associated with it, is an ordered sequence of itemsets or generalized itemsets $g_1, g_2, \ldots, g_n$ such that:*

- *if $sup(it, D_i) \geq min\_sup$ then g_i= it*
- *else g_i= git, where git is an ancestor of it, with respect to Γ, frequent in D_i and characterized by a minimum generalization level among the set of frequent ancestors of it, i.e., $git \in Anc[it, \Gamma] \wedge sup(git, D_i) \geq min\_sup$ and $\nexists$ $git_2 \in Anc[it, \Gamma]$ such that $L[git, \Gamma] > L[git_2, \Gamma]$ and $sup(git_2, D_i) \geq min\_sup$*

A HiGen HG_{it} may be represented as a sequence

$$g_1 \ \mathrm{R}_1 \ g_2 \ \mathrm{R}_2 \ \ldots \ g_{n-1} \ \mathrm{R}_{n-1} \ g_n$$

where R_i is a relationship holding between (generalized) itemsets g_i and g_{i+1} and may be represented as $\nearrow$ if g_i is a descendant of g_{i+1}, $\searrow$ if g_i is an ancestor of g_{i+1} or $\rightsquigarrow$ if $L[g_i, \Gamma] = L[g_{i+1}, \Gamma]$.

Table 6.2(b) reports the set of HiGens mined from the timestamped datasets D_1 and D_2 (See Tables 6.1(a) and 6.1(b)) by enforcing an absolute minimum support threshold equal to 2 and by exploiting the taxonomy reported in Figure 6.1. For instance, {*(location, Paris)*} $\nearrow$ {*(location, France)*} is a HiGen associated with itemset {*(location, Paris)*} given that {*(location, Paris)*} is a descendant of {*(location, France)*} and it is frequent in D_1 but infrequent in D_2. Consider now the time attribute in D_1 and D_2 (See Tables 6.1(a) and 6.1(b)) and the corresponding generalization hierarchy (see Figure 6.1(a)). {*(time, from 5 p.m. to 6 p.m.)*} $\searrow$ {*(time, 5 p.m.)*} is a HiGen associated with itemset {*(time, 5 p.m.)*} while {*(time, p.m.)*} $\searrow$ {*(time, 5 p.m.)*} does not as it exists an ancestor of {*(time, 5 p.m.)*}, i.e., {*(time, from 5 p.m. to 6 p.m.)*}, that is frequent in D_1 and such that L[{*(time, from 5 p.m. to 6 p.m.)*}, Γ] < L[{*(time, p.m.)*}, Γ].

6.1.2 The Non-redundant HiGen

Since a not generalized itemset *it* may have several ancestors characterized by the same generalization level, it trivially follows that an itemset may have

many associated HiGens. However, analysts may prefer to look into a more concise set of temporal change patterns instead of the whole HiGen set. The redundancy of an ancestor with respect to one of its descendants is defined in terms of their respective coverage sets. In the following, the concept of redundancy of an ancestor is formally stated.

Theorem 1 *The ancestors of a reference (generalized) itemset with minimal redundancy are the ones with minimal support.*

Proof 1 *Suppose that* X_1,$X_2 \in Anc[Y, \Gamma]$ *and* $red(X_1, Y) > red(X_2, Y)$. *Since* $|cov(X_1, D)| > |cov(X_2, D)|$ *it follows that* $sup(X_1, D) > sup(X_2, D)$.

Among the possible generalizations of an infrequent itemset belonging to the minimal abstraction level, the ones with minimal support are the generalizations that cover its knowledge with the minimal amount of redundancy (Cf. Theorem 1).

To reduce the number of considered patterns, the notion of Non-redundant HiGen is introduced. Among the set of HiGens relative to the same generalized itemset *it*, the Non-redundant HiGens are the HiGens that include generalized itemsets with minimal redundancy, i.e., the ones with minimal support, in case the reference itemset becomes infrequent in a certain time period.

Definition 26 Non-redundant HiGen**.** *Let* $\mathcal{D}$ *be an ordered sequence of timestamped structured datasets and* Γ *a taxonomy built over data items in* $D_i \in \mathcal{D}\ \forall\ i$. *A* Non-redundant HiGen SHG_{it}, *associated with it, is a* HiGen *composed of an ordered sequence of itemsets or generalized itemsets* $g_1, g_2, \ldots, g_n$ *such that for each generalized itemset* g_i *in* SHG_{it} *its redundancy with respect to it is minimal, i.e., it does not exist any frequent ancestor* g_i^* *of it such that* $sup(g_i^*, D_i) < sup(g_i, D_i)$.

From the above definition, it trivially follows that the number of Non-redundant HiGens associated with each not generalized itemset *it* is lower than the number of the corresponding HiGens. Consider, for instance, an item *(time, 5:35 p.m.)* and two of its possible generalizations *(time, from 5 p.m. to 6 p.m.)* and *(time, from 5:30 p.m. to 6:30 p.m.)* belonging to the same generalization level. Suppose that *(time, 5:35 p.m.)* $\nearrow$ *(time, from 5 p.m. to 6 p.m.)* and *(time, 5:35 p.m.)* $\nearrow$ *(time, from 5:30 p.m. to 6:30 p.m.)* are both HiGens, according to Definition 25. Among them,

the analyst may prefer the one that covers the same knowledge supported by the item *(time, 5:35 p.m.)* with a minimal amount of redundancy. If, for instance, the support of *(time, from 5 p.m. to 6 p.m.)* is strictly lower than the support of *(time, from 5:30 p.m. to 6:30 p.m.)* in the second time period the former HiGen is selected as Non-redundant HiGen.

Problem statement Given an ordered set of timestamped structured datasets, a taxonomy Γ, and a minimum support threshold *min_sup*, this paper addresses the problem of mining all HiGens, according to Definitions 25.

To efficiently accomplish the HiGen extraction task, in the next section the HiGen Miner (History GENeralized Pattern Miner) is presented. The main HiGen Miner algorithm modifications needed to address the Non-redundant HiGen extraction (Cf. Definition 26) are discussed as well.

6.2 The HiGen Miner Algorithm

The HiGen Miner (History GENeralized Pattern Miner) algorithm addresses the extraction of the HiGens, according to Definition 25.

HiGen mining may be addressed by means of a postprocessing step after performing the traditional generalized itemset mining step [103], constrained by the minimum support threshold and driven by the input taxonomy, from each timestamped dataset. However, this approach may become computationally expensive, especially at lower support thresholds, as it requires (i) generating all the possible item combinations by exhaustively evaluating the taxonomy, (ii) performing multiple taxonomy evaluations over the same pattern mined several times from different time periods, and (iii) selecting HiGens by means of a, possibly time-consuming, postprocessing step.

To address the above issues, a more efficient algorithm, called HiGen Miner, is proposed. It introduces the following expedients: (i) to avoid generating all the possible combinations, it adopts, similarly to GenIO [16], an Apriori-based support-driven generalized itemset mining approach, in which the generalization procedure is triggered on infrequent itemsets only. Unlike GenIO [16], the generalization process does not generate all possible ancestors of an infrequent itemset at any abstraction level, but it stops at the generalization level in which at least a frequent ancestor occurs, (ii) to prevent multiple taxonomy evaluations over the same pattern, the generalization process of each itemset is postponed after its support evaluation in

all timestamped datasets and iteratively applied on infrequent generalized itemsets of increasing generalization level, and (iii) to reduce the extraction time, HiGen generation is performed on-the-fly, without the need of an ad-hoc postprocessing step. Furthermore, a slightly modified version of the HiGen Miner algorithm is proposed to address Non-redundant HiGen extraction. A description of the main algorithm modifications is given in Section 6.2.2.

As a drawback, the HiGen Miner algorithm automatically selects the subset of generalized itemsets of interest at the cost of a higher number of dataset scans with respect to a traditional Apriori-based miner. Consider a timestamped datasets of n attributes and a taxonomy of height H_{max}, the HiGen Miner algorithm requires up to $H_{max} \cdot n$ dataset scans, while a traditional Apriori-like miner requires up to n dataset scans. However, traditional mining approaches still require a postprocessing step to select the HiGens of interest (Cf. Definition 25). The experimental evaluation, reported in Section 6.3, shows that the HiGen Miner algorithm yields good performance, in terms of both pattern pruning selectivity, with respect to previous generalized itemset mining approaches (i.e., [16, 103]), and execution time. In Section 6.2.1, a pseudo-code of the HiGen Miner is reported and thoroughly described while, in Section 6.2.2, the selection and categorization of the discovered HiGens is addressed.

6.2.1 The HiGen Miner algorithm pseudo-code

Algorithm 4 reports the pseudo-code of the HiGen Miner. The HiGen Miner algorithm iteratively extracts frequent generalized itemsets of increasing length from each timestamped dataset by following an Apriori-based level-wise approach and directly includes them into the HiGen set. At an arbitrary iteration k, HiGen Miner performs the following three steps: (i) k-itemset generation from each timestamped dataset in $\mathcal{D}$ (line 3), (ii) support counting and generalization of infrequent (generalized) k-itemsets of increasing generalization level (lines 6-37), (iii) generation of candidate itemsets of length $k+1$ by joining k-itemsets and infrequent candidate pruning (line 39). Since HiGen Miner discovers HiGens from relational datasets it adopts the well-known strategy to consider the structure of the input datasets to avoid generating combinations that do not comply with the relational data format. After being generated, frequent itemsets of length k are added to the corresponding HiGens in the HG set (line 9), while infrequent ones are generalized by means of the taxonomy evaluation procedure

(line 17). Given an infrequent itemset c of level l and a taxonomy Γ, the taxonomy evaluation procedure generates a set of generalized itemsets of level $l+1$ by applying, on each item $(t_j, value_j)$ of c, the corresponding generalization hierarchy $GH_j \in \Gamma$ (see Definition 3). All the itemsets obtained by replacing one or more items in c with their generalized versions of level $l+1$ are generated and included into the *Gen* set (line 21). Finally, generalized itemset supports are computed by performing a dataset scan (line 26). Frequent generalizations of an infrequent candidate c, characterized by level $l+1$, are first added to the corresponding HiGen set and then removed from the *Gen* set when their lower level infrequent descendants in each time period have been fully covered (lines 27- 32). In such a way, their further generalizations at higher abstraction levels are prevented. Notice that the taxonomy evaluation over an arbitrary candidate of length k is postponed when the support of all candidates of length k and generalization level l in each timestamped dataset is available. This approach allows triggering the generalization on distinct candidates of level l that are infrequent in at least one timestamped dataset in $\mathcal{D}$. The sequence of support values of an itemset that is infrequent in a given time period is store and reported provided that (i) it has at least a frequent generalization in the same time period, and (ii) it is frequent in at least one of the remaining time periods. The generalization procedure stops, at a certain level, when the *Gen* set is empty, i.e., when either the taxonomy evaluation procedure does not generate any new generalization or all the considered generalizations are frequent in each time period and, thus, have been pruned (line 30) to prevent further knowledge aggregations.

The HiGen Miner algorithm ends the mining loop when the set of candidate itemsets is empty (line 40).

6.2.2 HiGen categorization and selection

Domain experts are commonly in charge of looking into the discovered temporal change patterns to highlight most notable trends. To ease the domain expert validation task, a preliminary analysis of the set of extracted HiGens may (i) categorize them based on their time-related trends, or (ii) prune them to select a subset of interest, i.e., the Non-redundant HiGens.

HiGen *categorization:* To better highlight HiGen significance, HiGens are categorized, based on their time-related trend, in: (i) *Stable* HiGen*s*, i.e., HiGens that include generalized itemsets belonging to the same generalization level, (ii) *monotonous* HiGen*s*, i.e., HiGens that include a se-

quence of generalized itemsets whose generalization level shows a monotonous trend, and (iii) *oscillatory* HiGENs, i.e., HiGENs that include a sequence of generalized itemsets whose generalization level shows a variable and non-monotonous trend. Since, according to Definition 9, a generalized itemset of level l may have several generalizations of level $l + 1$ and taxonomies may have unbalanced data item distributions, *stable* HiGENs may be further partitioned in: (i) *Strongly stable* HiGENs, i.e., stable HiGENs, in which items, appearing in its generalized itemsets and belonging to same data attribute, are characterized by the same generalization level, and (ii) *Weakly stable* HiGENs, i.e., stable HiGENs in which items, appearing in its generalized itemsets and belonging to the same attribute, may be characterized by different generalization levels. Examples of HiGENs, extracted from a real-life context dataset, belonging to each category are reported in Section 6.4.

Non-redundant HiGen *selection:* To reduce the amount of generated patterns, analysts may focus on the set of Non-redundant HiGENs (Cf. Definition 26). Since they represent the subset of HiGENs whose generalizations cover the same knowledge covered by the reference infrequent itemset with minimal redundancy, they may be deemed relevant by domain experts. Notice that the extraction of the Non-redundant HiGENs may be accomplished by slightly modifying the HiGen Miner algorithm (see Algorithm 4). A sketch of the main algorithm modifications follows. At each algorithm iteration, once the the list of *Gen* set is populated, the set of infrequent descendants of patterns in *Gen* is visited. A nested loop iterates on *Gen* in order of increasing support and selects its generalizations with minimal support. Finally, the set HG of discovered temporal change patterns is updated accordingly.

6.3 HiGen Miner performance analysis

This section evaluates the HiGen Miner (History GENeralized Pattern Miner) algorithm by means of a large set of experiments addressing the following issues: (i) the pruning selectivity, in terms of the number of generalized itemsets extracted by HiGen Miner with respect to previous generalized itemset mining algorithms, i.e., [16, 103] (Section 6.3.2), (ii) the performance, in terms of the number of extracted temporal patterns, of the HiGen Miner algorithm (Section 6.3.3), and (iv) the scalability, in terms of execution time, of the proposed approach (Section 6.3.4).

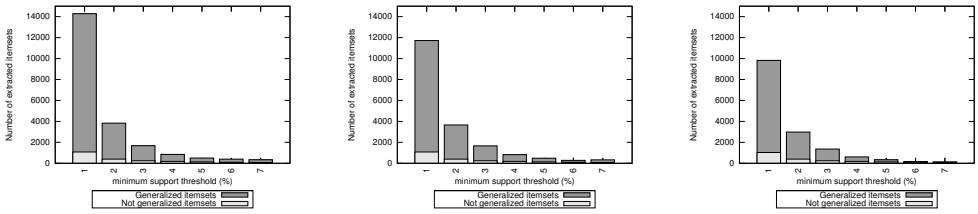
All the experiments were performed on a 3.2 GHz Pentium IV system

with 2 GB RAM, running Ubuntu 8.04. The HiGen Miner algorithm was implemented in the Python programming language [95].

The evaluation of the pruning selectivity of the HiGen Miner generalization procedure is performed on synthetic datasets. It compares the number of frequent itemset and generalized itemsets mined from each generated timestamped dataset with that extracted by the following generalized frequent itemset mining algorithms: (i) a traditional algorithm, i.e., Cumulate [103], which performs an exhaustive taxonomy evaluation by generating all possible frequent combinations of generalized and not generalized itemsets, and (ii) a recently proposed support-driven approach to itemset generalization, i.e., GenIO [16], which generates a generalized itemset only if it has at least an infrequent descendant. The set of experiments was performed on synthetic datasets generated by means of the TPC-H generator [111], whose main configuration settings follow.

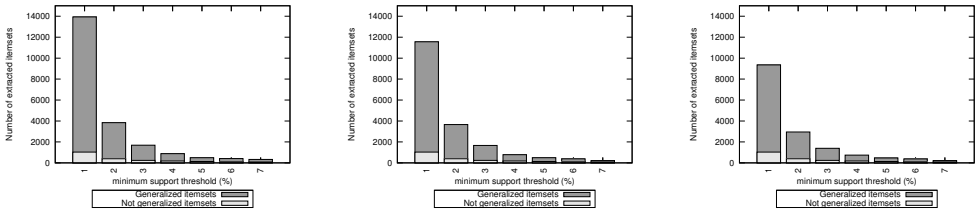
6.3.1 Synthetic datasets

The TPC-H data generator [111] consists of a suite of business-oriented ad-hoc database queries. By varying the scale factor parameter, files with different size could be generated. Generalized itemsets have been mined from a dataset generated from the lineitem table by setting a scale factor equal to 0.075 (i.e., around 450,000 records). Hierarchies on line item categorical attributes have been generated by using the part, nation, and region tables. To generate generalization hierarchies over the continuous attributes, several data equi-depth discretization steps of finer granularities are applied. The finest discretized values are considered as data item values and, thus, become the taxonomy leaves, while the coarser discretization procedures are exploited to aggregate the corresponding lower level values. More specifically, pairs of consecutive discretized values are aggregated in higher level items. To partition the whole dataset in three distinct time-related data collections the source data is queried by enforcing different constraints on the shipping date value (attribute *ShipDate*). More specifically, line items shipped in the three following time periods are partitioned: [1992-01-01, 1994-02-31], [1994-03-01, 1996-05-31] [1996-06-01, 1998-12-01]. For the sake of brevity, they will denote the corresponding datasets as data-1, data-2, and data-3 in the rest of this section.



(a) Cumulate (b) GenIO (c) HiGen Miner

Figure 6.2: HiGen Miner: generalized and not generalized itemsets extracted from data-1



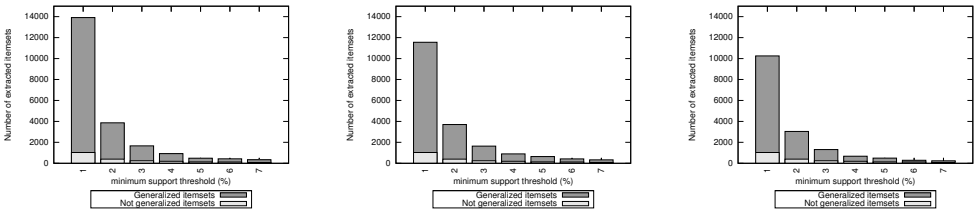
(a) Cumulate (b) GenIO (c) HiGen Miner

Figure 6.3: HiGen Miner: generalized and not generalized itemsets extracted from data-2

6.3.2 Performance comparison

Since the enforcement of the minimum support threshold during the itemset mining step significantly affects the number of extracted itemsets, I performed different mining sessions, for all combinations of algorithms and datasets, by varying the minimum support threshold value. In Figures 6.2, 6.3, and 6.4 is plotted the number of itemsets mined, respectively, from data-1, data-2, and data-3. To test the HiGen Miner algorithm, the generated datasets are considered in order of increasing shipment date interval. To better highlight the pruning selectivity on the cardinality of the mined generalized itemsets, it distinguished between generalized itemsets and not.

For all the tested algorithms and datasets and for most of the settings of minimum support value, the percentage of extracted frequent itemsets including at least one generalized item is significant (i.e., higher than 60%). For both Cumulate and GenIO, the number of mined (generalized) itemsets significantly increases when lower minimum support values are enforced



(a) Cumulate (b) GenIO (c) HiGen Miner

Figure 6.4: HiGen Miner: generalized and not generalized itemsets extracted from data-3

(e.g., 1%). Thus, the analysis of the temporal evolution of the extracted itemsets may require a computationally expensive postpruning step. In GenIO, infrequent items are aggregated during the extraction process and all frequent ancestors of an infrequent itemset are extracted. However, most of the extracted generalization seem redundant as they represent the same infrequent knowledge at increasing abstraction levels. The less redundant generalizations could be selected as the less redundant frequent representatives. By following this approach, the HiGen Miner algorithm selects just the frequent ancestors with minimal generalization level (i.e., with minimal redundancy). The pruning selectivity, in terms of the number of extracted generalized itemsets, achieved by HiGen Miner within each time period appears more evident when lower support thresholds (e.g., 2%) are enforced (see Figures 6.2(c), 6.3(c), and 6.4(c)). In fact, when high support thresholds (e.g., 7%) are enforced, most of the frequent generalizations have already a high generalization level and, thus, the pruning effectiveness is less evident. Oppositely, when lower support thresholds are enforced (e.g., 2%), the extraction of a significant amount of redundant generalized itemsets, generated by both Cumulate and GenIO, is prevented. The pruning rate on the number of mined higher level (generalized) itemsets is between 5% and 15% with respect to GenIO for any support threshold value.

6.3.3 HiGen Miner parameter analysis

This section analyzed the performance of the HiGen Miner algorithm, in terms of the number extracted HiGens and Non-redundant HiGens, by analyzing the impact of the following factors: (i) minimum support threshold, (ii) number of time periods, and (iii) taxonomy characteristics. A set of

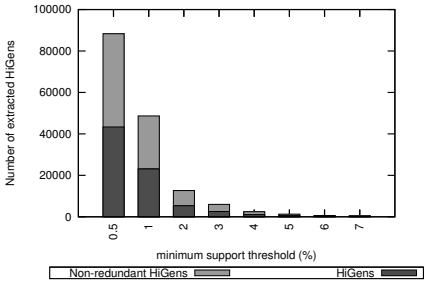
experiments was performed on synthetic datasets generated by means of the TPC-H generator [111]. The baseline configuration used for data generation is similar to that described in Section 6.3.1.

Impact of the support threshold The impact of the minimum support threshold on the number of discovered HIGENs and NON-REDUNDANT HIGENs is analyzed on synthetic datasets. In Figure 6.5(a) is reported the number of HIGENs mined from data-1, data-2, and data-3 by varying the minimum support threshold. To test the HIGEN MINER algorithm, datasets are considered in order of increasing shipment date interval. To better highlight the pruning selectivity yielded by mining NON-REDUNDANT HIGENs, it distinguished between NON-REDUNDANT HIGENs and not.

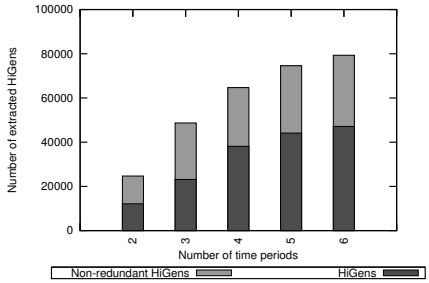
As expected, the number of generated HIGENs increases more than linearly when the minimum support threshold decreases. The selection of the NON-REDUNDANT HIGENs significantly reduces the number of generated temporal change patterns. In fact, the number of possible generalizations with minimal generalization level of each infrequent itemset significantly affects the cardinality of the set of mined HIGENs. Let *min_sup* be a minimum support threshold. Let n be the number of considered time periods (i.e., the number of timestamped datasets) and let $|\{p_i\}|$ be the average number of not generalized patterns p_i that are frequent, with respect to *min_sup*, in an arbitrary time period at any abstraction level. Let *avg_level_sharing* be the average number of frequent generalizations, with minimum generalization level, of an infrequent pattern. An estimation of the number of HIGENs ($|HiGen|$), mined by enforcing a support threshold *min_sup*, is:

$$|HiGen| \approx avg\_level\_sharing \cdot (n-1) \cdot |\{p_i\}| \tag{6.1}$$

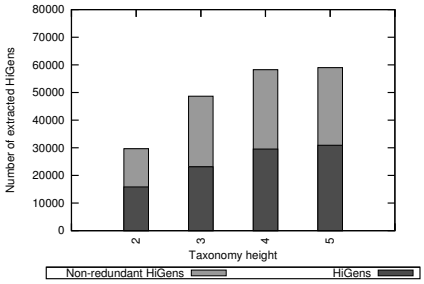
By setting $n = 3$ and by approximating the $\{p_i\}$ set cardinality in Formula 6.1 as the greatest common divisor of the number of generalized itemsets extracted from data-1, data-2, and data-3, for each support threshold (see Figures 6.2(c), 6.3(c), and 6.4(c)), The approximated value achieved by *avg_level_sharing* could be computed, for any support threshold, starting from the results reported in Figure 6.5(a). The average number *avg_level_sharing* of frequent generalizations, with minimal generalization level, of each infrequent itemset turns out to range from 2 to 4 for any support threshold. The achieved results are close to the expectations. By selecting the NON-REDUNDANT HIGENs, the reduction, in terms of the number of HIGENs, is greater than 35% for any support threshold. The obtained results depend on the considered data item distribution.



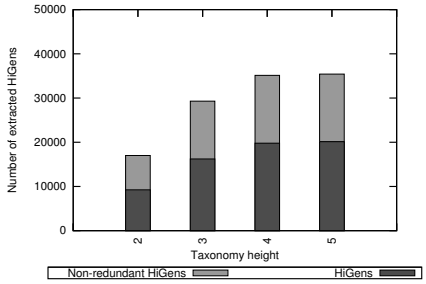
(a) Impact of the support threshold. Time periods=3. Taxonomy height=3



(b) Impact of the number of time periods. *min_sup*=1%. Taxonomy height=3



(c) Impact of the taxonomy height. Equi-depth discretization. *min_sup*=1%. Time periods=3.



(d) Impact of the taxonomy height. Entropy-based discretization. *min_sup*=1%. Time periods=3.

Figure 6.5: HiGen Miner: number of mined HiGens and Non-redundant HiGens.

Impact of the number of time periods The impact of the number of considered time periods on the cardinality of the extracted patterns is also analyzed. To this aim, the *ShipDate* attribute domain of the lineitem table is partitioned in an increasing number of intervals. Figure 6.5(b) reports the number of mined HiGens and Non-redundant HiGens. The cardinality of the discovered HiGens grows when the number of time periods increases due to both the presence of new reference itemsets that are frequent, at any abstraction level, in at least one time period and the generation of multiple combinations of least generalized itemsets. As expected, the increase is less relevant when considering just Non-redundant HiGens.

Impact of the taxonomy characteristics The impact of the main taxonomy characteristics on the number of extracted HiGens and Non-redundant HiGens is analyzed on synthetic datasets as well. To evaluate the impact of the taxonomy height, ad-hoc taxonomies of increasing height are generated by varying the number of discretization steps adopted for aggregating continuous data attribute values. To also evaluate the impact of different multiple-level data item distributions, Figures 6.5(c) and 6.5(d) report the number of extracted temporal patterns, when varying the taxonomy height, by applying two representative discretization procedures, i.e., an equi-depth procedure [107] and an entropy-based one [48], on numeric data attributes. A similar trend is shown by both the tested item distributions. By using a taxonomy composed of only 2 levels many generalizations are still infrequent in some time periods and, thus, the corresponding HiGens are not extracted. When the taxonomy height increases from 2 to 4 a significant number of reference infrequent itemsets becomes frequent at a higher generalization level and, thus, the number of extracted combinations relevantly grows. Further taxonomy height increments do not produce any significant increase of the HiGen cardinality. Since the data distribution generated by the entropy-based discretization is sparser than that generated by the equi-depth procedure, it produces a smaller number of temporal correlations.

6.3.4 HiGen Miner scalability

The scalability of the HiGen Miner algorithm, in terms of the execution time is evaluated on synthetic datasets. To analyze the scalability with the number of dataset records, the TPC-H data generator [111] is exploited by varying the scale factor to generate datasets of increasing size, ranging from $150,000$ to $900,000$ records. To test the HiGen Miner, the same configurations for the data generator already described in Section 6.3.1 is exploited. Figure 6.6 plots the time spent in HiGen mining for different support values. The computational complexity significantly increases for lower minimum support threshold values (e.g., 1%) due to the higher number of extracted itemsets. However, the execution time scales roughly linearly with the number of dataset records for any support threshold.

The scalability of the HiGen Miner with both the taxonomy height and the number of time periods is also analyzed. Results show that HiGen Miner averagely scales more than linearly with both taxonomy height and number of time periods due to the non-linear growth of the number of considered (potentially relevant) combinations. However, the HiGen Miner

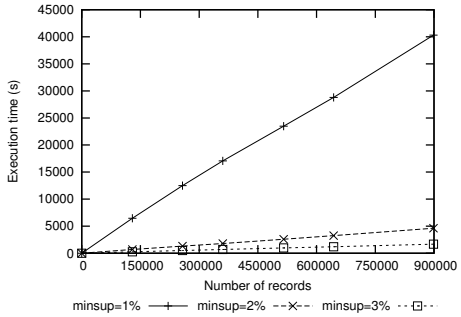


Figure 6.6: HiGen Miner algorithm scalability. Time periods = 3. Taxonomy height = 3

execution time is still acceptable even when coping with quite complex taxonomies evaluated over a number of time periods (e.g., around 20,000 seconds with *min_sup*=1%, time periods = 6, taxonomy height = 5, number of records = 300,000).

6.4 HiGen Miner application to context-aware data

To validate the significance of the results achieved by means of the HiGen Miner (History GENeralized Pattern Miner) algorithm, a campaign of experiments has been conducted on a real-life dataset coming from a context-aware mobile system. The extracted HiGENs have been first processed, based on the procedure described in Section 6.2.2. Next, they have been analyzed by a domain expert to assess their usefulness in the context of mobile user and service profiling.

This Section is organized as follows. Section 6.4 provides a brief description of the adopted dataset. Section 6.4 reports a selection of the discovered HiGENs while Section 6.4 describes the characteristics of the discovered HiGENs based on the proposed categorization.

Real context dataset

A real dataset, called *mDesktop*, is provided by a research hub, namely the Telecom Italia Lab. It collects contextual information about user application requests submitted, through mobile devices, over the time period of three

months (i.e., from August to October). From the whole context data collection, I generated 3 different timestamped datasets, each one corresponding to a 1-month time period. Regarding privacy concerns related to real context data usage, please notice that (i) experimental data were collected from voluntary users that provide their whole informed consent on personal data treatment for this research project, and (ii) real user names were hidden throughout the paper to preserve identities. A more detailed description of the adopted data collection follows.

mDesktop dataset. The Telecom Italia mobile desktop application provides a wide range of services to users through mobile devices. Some of them provide recommendations to users on restaurants, museums, movies, and other entertainment activities. For instance, they allow end users to request for a recommendation (GET_REC service), enter a score ($VOTE$ service), or update a score for an entertainment center ($UPDATE$ service). Other services allow users to upload files, photos, or videos and share them with the other users. The dataset includes 1,197 user requests concerning file sharing and uploading, 5814 records concerning user recommendations, and 4487 records regarding other kinds of services (e.g., weather forecasts, SMS, and call requests). To perform HiGen mining, a taxonomy including the following generalization hierarchies has been defined.

- **time stamp** → hour → time slot (two-hour time slots) → time slot (six-hour time slots) → day period (AM/PM)
- **service** → service category
- **latitude:longitude** → city → country
- **phone number** → call type (PERSONAL/BUSINESS)

In particular, service characterization is performed by exploiting the generalization hierarchy over the *Service* attribute domain reported in Figure 6.7.

The experimental campaign also considered (i) different time periods, and (ii) different generalization hierarchies for the *time stamp* attribute. In particular, it also considered 1-week time periods and 4-hour and 8-hour time slots. Some remarkable examples of HiGens mined by using these configurations are reported in Section 6.4.

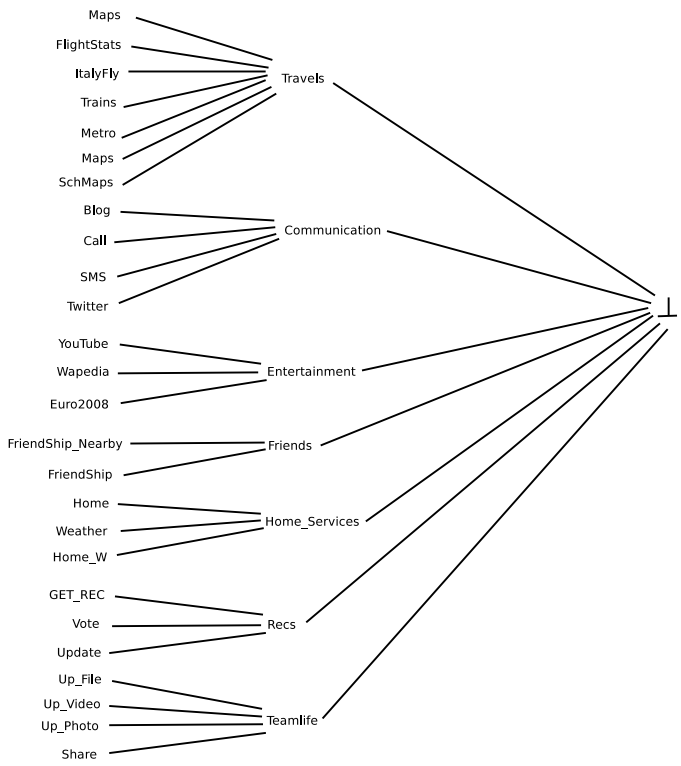


Figure 6.7: Generalization hierarchy for the *Service* attribute of the context dataset

Discovered pattern analysis

In the following, a selection of the discovered HiGens is reported. Furthermore, possible scenarios of usage for the selected patterns suggested by the expert are discussed. The reported HiGens are classified based on the categorization presented in Section 6.2.2. Note that all the reported HiGens, selected by the expert as most notable patterns, are also Non-redundant HiGens. Indeed, the pruning of the not Non-redundant HiGens did not relevantly affect the effectiveness of the knowledge discovery process.

Examples of stable HiGens *Stable* HiGens are HiGens whose generalized itemsets have the generalization level in common. Table 6.3 reports two examples of extracted strongly stable HiGens, namely the HiGens 1 and 2, and one example of weakly stable HiGen, namely the HiGen 3, generated by enforcing a minimum support threshold $min\_sup = 0.001\%$.

Both the reported strongly stable HiGENs concern the usage time of service *Twitter*, which provides mobile access to a famous online community. Since in HiGENs 1 and 2 all the items belonging to attribute *Service* are characterized by generalization level 1, while the ones belonging to attribute *Time* belong, respectively, to levels 2 and 3, the considered HiGENs are categorized as strongly stable HiGENs. The first strongly stable HiGEN states that requests for *Twitter* are frequently submitted between 9 p.m. and 10 p.m. within each considered month. The second one states a similar recurrence at a higher temporal abstraction level, i.e., between 7 p.m. and 13 p.m.. Their joint extraction implies that service *Twitter* is frequently requested from 9 p.m. to 10 p.m. within each month, but exists at least one hourly time slot between 7 p.m. and 13 p.m., different from [9 p.m., 10 p.m.], such that is infrequent within each considered month. I also analyzed the recurrences in service *Twitter* usage mined from shorter time periods. Even by considering weekly time periods, analogous trends are shown. Differently, the third stable HiGEN concerns the usage time of the whole application service by a user called John. Although, according to Definition 9, all its generalized itemsets have the same generalization level (i.e., 3), it is categorized as weakly stable HiGEN provided that the level of the items belonging to attribute *User* changes from month to month. This means that the information stored in the generalized itemsets changes its abstraction level, with respect to the input taxonomy, for a certain extent. However, since the overall itemset generalization level is affected by the mostly generalized attribute values (i.e., values associated with attribute *Time*), whose level does not vary over time, it still retains a certain degree of stability.

The information provided by stable HiGENs may be deemed relevant for service provisioning. For instance, the analyst may profitably exploit this knowledge for suiting bandwidth shaping to the actual user needs. In particular, he may either allocate dedicated bandwidth to frequently requested services at specific time slots or use free bandwidth at less congested time slots for network monitoring purposes.

Examples of monotonous HiGENs *Monotonous* HiGEN s are HiGENs whose sequence of generalized itemsets is characterized by generalization levels with monotonous trend due to a steadfast and significant decrease/increase of the itemset support values, with respect to the support threshold, in consecutive time periods. Consider the monotonous HiGENs reported in Table 6.3, which have been generated by enforcing a minimum support threshold $min\_sup = 0.001\%$.

Consider service *Home_W* belonging to category *Home_Services*, which is targeted to provide facilities to users who are at home. The first monotonous HiGen (i.e., HiGen 4) shows a recurrence in user John *mDesktop* application service usage. Requests for service *Home_W* are frequently submitted in August, while become infrequent in both September ($sup = 0.0001\%$) and October ($sup = 0.0008\%$). Nevertheless, the corresponding category *Home_Services* remains frequent in all the considered months. Readers can notice that weak monotonicity holds as the corresponding itemset generalization level increases from August to September while remains constant from September to October. It is worth mentioning that, by considering weekly time periods separately within each month, the reference itemset {*(Service, Home_W), (User, John)*} generates a strongly stable HiGen over August, while it generates oscillatory HiGens in the next months. Differently, the monotonous HiGen 5 highlights an opposite trend. It regards service *Trains*, which belongs to service category *Travels* and provides information on rail transports. User John appears less interested in service *Trains* in August ($sup = 0.0002\%$), possibly due to holiday vacations, while the service increases its attractiveness in the following months.

The above information is deemed relevant by domain expert for both user and service profiling. The analyst could (i) shape service bandwidth depending on the time period, (ii) personalize user *John* service promotions, and (iii) plan long-term promotions to counteract service usage decrease at certain times of the year. The extracted information may be also used for cross-selling purposes, i.e., by suggesting others services belonging to category *Home_Services* (e.g., service *Weather*) to either increase user interest in the service category or counteract user migration from occasional use services (i.e., classes of services, such as $Travels$, on which users focus their interest for short time periods) to commonly used services (e.g., service $Home\_W$).

Examples of oscillatory HiGens *Oscillatory* HiGens are HiGens whose sequence of generalized itemsets is characterized by generalization levels with variable and non-monotonous trend. Consider, for instance, the oscillatory HiGens, reported in Table 6.3, mined by enforcing a minimum support threshold equal to 0.01%.

The first oscillatory HiGen (i.e., HiGen 6) concerns service category $Travels$, which provides to users information about travels, and its service $Maps$, which allows browsing of geographical maps. User Terry seems interested in services belonging to category $Travels$ in each considered month.

Nevertheless, in September, he focused his interest in service *Maps*, possibly due to work travels. Differently, the second oscillatory HiGen (i.e., HiGen 7) provides information about the usage time of service MMS. In August and October, the service MMS is frequently requested between 13 p.m. and 14 p.m., while in September user interest in service MMS in the same time slot decreases but still holds between 13 p.m. and 15 p.m. Notice that, by using a different 2-hour time stamp (between 12 p.m. and 14 p.m.) the temporal correlation does not hold anymore. By looking into the discovered oscillatory HiGens, analysts may, for instance, investigate the provision of occasional use services (e.g., service *Maps*) and perform the following actions: (i) suggest complementary services (e.g., service *Flightstats*), (ii) plan promotions targeted to specific user profiles, and (iii) discover mostly used service parameters to automatically suggest (e.g., frequently requested GPS coordinates for service *Maps*).

Characteristics of the discovered HiGens

The expert also analyzes the frequency of stable, monotonous, and oscillatory HiGens inherent in a certain service category. To address this issue, he adopts the following three-step approach: (i) HiGen mining from *mDesktop* by means of the HiGen Miner algorithm, by enforcing a minimum support threshold $min\_sup = 0.001\%$, (ii) selection of the HiGens whose generalized itemsets include items belonging to the *Service* attribute (i.e., service description), and (iii) HiGen categorization based on the service categories reported in Figure 6.7. For instance, the monotonous HiGen $\{(Service, SMS)\} \nearrow \{(Service, Communication)\}$ would be associated with category *Communication*. Table 6.4 reports the distribution of the HiGens within the service categories.

Categories that include commonly used services (e.g., category *Communication*) are mainly represented by stable HiGens (e.g., 52% of the HiGens belonging to *Communication* are stable against 22% of monotonous HiGens and 26% of oscillatory HiGens), while categories that group occasional use services are mostly covered by monotonous or oscillatory HiGens. A significant percentage of strongly stable HiGens characterizes service categories with a great regularity in their context of use (e.g., category *Home_Services*). For these services, analysts may design long-term resource and promotion plans. Service categories with a significant percentage of weakly stable or monotonous HiGens (e.g., *Entertainment* and *Friends*) are usually suitable for medium-term plans, as they show periodical (e.g., seasonal) variations of their context of usage. Finally, oscillatory HiGens are often characterized

by non-deterministic behaviors. For instance, notice that service category *Travels* includes a significant percentage of oscillatory HiGens as its usage changes significantly and unexpectedly from month to month.

6.5 HiGen Miner application to social data: the DyCoM Framework

This section presents the DyCoM (Dynamic COntext Miner) data mining system. It focuses on discovering and representing dynamic higher level correlations among data posted by users on the Twitter microblogging Web site in the form of dynamic association rules. Their extraction is based on the HiGen Miner algorithm. By exploiting the Twitter Application Programming Interface (API), DyCoM retrieves both the tweet content (i.e., the textual messages) and their contextual features (e.g., publication date, time, place). Based on the values of one of the most peculiar tweet contextual features (e.g., the publication date), tweets are first partitioned in an ordered sequence of tweet collections. The textual content is tailored to a relational data schema [108] to enable the association rule mining process. Next, a taxonomy over contextual data features is semi-automatically built, by means of Extraction, Transformation, and Loading (ETL) processes, and exploited to drive the rule mining process from the sequence of tweet collections. Generalized association rules are extracted by aggregating, according to the taxonomy, lower level data items into higher level ones. The discovered correlations provide a higher level view of the analyzed data since they also include feature values (e.g., places, time stamps) at different levels of abstraction (e.g., regions or nations, time intervals or days). DyCoM integrates the discovery of generalized association rules in the context of dynamic rule mining. The discovered patterns compactly represent higher level correlations and their temporal evolution across a sequence of tweet collections.

Finally, the DyCoM framework allows effectively querying the extracted patterns, based on either their schema or content, to discover recurrences hidden in Web user behaviors and topic trends. For instance, significant correlations regarding specific topics may be pointed out in consecutive days at different geographical granularity levels, e.g., within a certain city or its corresponding region or nation.

This section is organized as follows. A more though description of the adopted framework is reported in Section 6.5.1, while, in Section 6.5.2, two examples of real-life use-cases for the DyCoM system are presented.

6.5.1 Characteristics of the DyCoM Framework

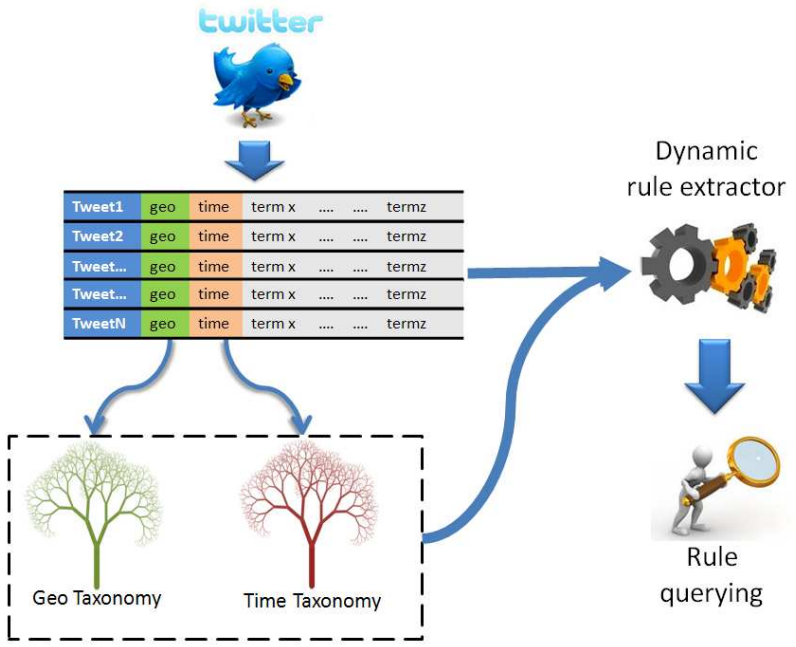


Figure 6.8: The DyCoM framework

DyCoM (Dynamic Context Miner) is a data mining framework focused on supporting the discovery of relevant correlations among the textual content and the publication context of messages posted on Twitter. To address this issue, it investigates the evolution of most significant patterns hidden in a sequence of tweet collections. Figure 6.8 reports the DyCoM framework architecture. In the following, its main blocks and functionalities are briefly described.

User-generated content and context data retrieval. This block entails the retrieval, preprocessing, and categorization of messages (tweets) posted by Web users on Twitter. The retrieved data is tailored to a relational data schema that includes both content features (i.e., the most relevant keywords) and contextual features (e.g., the geographical location). Furthermore, tweets are partitioned into a sequence of subsets based on the value its most peculiar features (e.g., tweet collections posted in consecutive days).

Taxonomy generation over contextual data. Taxonomies over the tweet contextual features are semi-automatically generated. More specifically, Ex-

traction, Loading, and Transformation (ETL) processes are exploited to aggregate values of lower level contextual features (e.g., the GPS coordinate) into their higher level aggregations (e.g., the city and the region).

Dynamic rule extractor. This block aims at discovering the evolution of the significant higher level correlations hidden in the sequence of tweet collections. It discovers dynamic patterns, in the form of dynamic generalized association rules, whose main quality indexes (i.e., support and confidence) exceed a given threshold and possibly vary from one interval to another. The generalization is performed by evaluating the previously generated taxonomies. Their extraction is addressed by means the commonly used two-step process: (i) dynamic itemset mining, driven by the support threshold and (ii) dynamic association rule generation, driven by the confidence constraint. The first mining step is based on the HiGen Miner algorithm.

Rule querying. The extracted dynamic rules are queried to efficiently retrieve the information of interest based on either their content or schema. To ease the domain expert analysis' task, the resulting dynamic rules are ranked based on the value of their main quality indexes (e.g., support and confidence) in the user-provided tweet collections of increasing level.

6.5.2 Examples of DyCoM use-cases

In this section, two real use-cases for the DyCoM system that are focused on user behavior and topic trend analysis are presented. For each use-case, some examples of discovered dynamic generalized association rules are reported as well.

Use-case 1: Topic trend analysis

This application scenario enables analysts to look into newsworthy topic trends by analyzing the temporal evolution of correlations hidden in the tweet collections. To this aim, analysts may follow this steps: (i) crawling collections of tweets posted at consecutive time periods, (ii) dynamic generalized association rule mining from tweet collections in order of increasing time periods, and (iii) rule querying based on user-specified constraints.

Tweet contextual features include the time stamp at which messages are posted. This information is exploited to partition tweets into disjoint collections associated with different time periods (e.g., 1-day time period). The

dynamic mining algorithm allows discovering high level recurrences in the form of generalized association rules. They may represent unexpected trends in the evolution of relevant tweet topics. For instance, analysts may wonder how breaking news are matter of contention on Twitter. To delve into the impact of breaking news coming from the United States Capitol, tweets whose submission time is uniformly distributed in the range [2011/03/22, 2011/03/24] are first collected. Then, they are categorized based on their submission date. For example, the following dynamic generalized rules are extracted by enforcing a minimum support threshold equal to 1%.

1. {(Keyword, Obama),(Keyword2, Libya)} → {(Place, Washington, D.C.)} (sup=[1.3%, 0.3%, 0.3%], , conf=[100%, 83%, 85%])
2. {(Keyword, Obama),(Keyword2, Libya)} → {(Place, Washington, D.C.)} (sup=[2.3%, 2.5%, 1.3%], , conf=[100%, 91%, 91%])

The U.S. congress meeting, that held on March 22nd in Washington D.C., was focused on the conflict in Libya. Keywords *Obama* and *Libya*, which have been frequently posted on March, 22nd in Washington, D.C., becomes infrequent in the same location the day after. However, the extraction of the higher level correlation (B) allows figuring out that the same topic remains of interest in the U.S.A. yet.

Use-case 2: Context-based user behavior analysis

This application scenario investigates the attitudes of Twitter users in posting, citing, and answering Twitter messages concerning newsworthy topics in different spatial contexts. To achieve this goal, analysts should perform the following steps: (i) crawling tweet collections posted from regions or states of interest within a given time period, (ii) dynamic generalized association rule mining from tweet collections by following a user-provided significance order, and (iii) dynamic rule querying based on user-specified constraints.

The previously collected tweets, posted during the time period [2011/03/22, 2011/03/24], are considered and reorganized, based on their submission place, in: (i) a collection of tweets posted within a $2,500$ km radius far from New York (i.e., lands along Eastern American coastline) and (ii) a collection of tweets posted within a $2,500$ km radius far from London (i.e., North-West of Europe). A dynamic generalized association rule mining session is performed

by enforcing a minimum support threshold equal to 1% and a minimum confidence threshold equal to 80% and by considering the American tweet collection first. Following the chain of spatial tweet message propagation set the analyst may discover valuable knowledge about user attitudes in Twitter service usage by looking into the history of the discovered patterns across a sequence of places of interest. For instance, the following dynamic rules are extracted:

1. {(Keyword, Obama), (Place, Washington, D.C.)} → {(Date, 2011/03/22)} (sup=[3.6%, 2.1%, 0.5%], conf=[100%, 100%, 91%])

2. {(Keyword, Obama), (Place, United Kingdom)} → {(Date, 2011/03/22)} (sup=[1.1%, 2.3%, 1.2%], [conf=100%, 95%, 95%])

The dynamic generalized association rules (A) is discovered from the collection of American tweets, while rule (B) is mined from the European tweet collection. American Twitter users paid particular attention to the foreign policy undertaken by president Obama and the American Congress, which had been topic of discussion in the past meeting held in the United States Capitol Washington, D.C. (USA) on March, 22nd 2011. Furthermore, users coming from Europe are also interested in posting messages related to the same topic as the U.S. Government decisions bias the European state economies.

Not generalized itemset	**Sup** D_1	**Sup** D_2
{Turin}	1 (Inf.)	2
{Paris}	2	1 (Inf.)
{T-shirt}	2	2
{Jacket}	3	3
{T-shirt, Turin}	1 (Inf.)	2
{Jacket, Paris}	2	1 (Inf.)
Generalized itemset	**Sup** D_1	**Sup** D_2
{Italy}	2	3
{France}	3	2
{T-shirt, Italy}	2	2
{Jacket, France}	3	2

(a) Generalized and not generalized itemsets mined from D_1 and D_2.

HIGENS **from** D_1 **to** D_2
{T-shirt} [sup = 2] $\rightsquigarrow$ {T-shirt} [sup = 2]
{Jacket} [sup = 3] $\rightsquigarrow$ {Jacket} [sup = 3]
{Paris} [sup = 2] $\nearrow$ {France} [sup = 2]
{T-shirt, Italy} [sup = 2] $\searrow$ **{T-shirt, Turin}** [sup = 2]
{Jacket, Paris} [sup = 2] $\nearrow$ {Jacket, France} [sup = 2]
{Italy} [sup = 3] $\searrow$ **{Turin}** [sup = 2]

(b) Extracted HIGENS.

Table 6.2: HIGEN MINER: Extracted patterns. $min\_sup = 2$.

Algorithm 4 HiGen Miner: History GENeralized Pattern Miner

Input: ordered set of timestamped structured datasets $\mathcal{D}$=$\{D_1, D_2, \ldots, D_n\}$, minimum support threshold *min_sup*, taxonomy Γ
Output: set of HiGens HG

$k = 1$ // Candidate length
$HG = \emptyset$
C_k = set of distinct k-itemsets in D
repeat
  for all $c \in C_k$ **do**
    scan D_i and count the support sup(c,D_i) $\forall$ $D_i \in \mathcal{D}$
  end for
  $L_k^i = \{$itemsets $c \in C_k |$ sup(c,D_i) $\geq$ *min_sup* for some $D_i \in D\}$
  HG = update_HiGen_set(L_k^i, HG)
  $l = 1$ // Candidate generalization level
  $Gen = \emptyset$ // generalized itemset container
  repeat
    for all c in C_k of level l **do**
      $D_c^{\inf} = \{D_i \in D \mid$ sup(c,D_i) $<$ *min_sup*$\}$ // datasets for which c is infrequent
      if $D_c^{\inf} \neq \emptyset$ **then**
        $gen(c)$ = set of new generalizations of itemset c of level $l+1$
        $gen(c)$ = taxonomy_evaluation(Θ, c)
        for all $gen \in gen(c)$ **do**
          $gen.Desc = c$ // Generalized itemset descendant
        end for
        $Gen = Gen \cup gen(c)$
      end if
    end for
    if $Gen \neq \emptyset$ **then**
      for all $gen \in Gen$ **do**
        scan D_i and count the support of gen $\forall$ $D_i \in D_{gen.Desc}^{\inf}$
        for all $gen|$ sup(gen,D_i) $\geq$ *min_sup* for some $D_i \in D_{gen.Desc}^{\inf}$ **do**
          HG = update_HiGen_set(gen, HG)
          if sup(gen,D_i) $\geq$ *min_sup* for all $D_i \in D_{gen.Desc}^{\inf}$ **then**
            remove gen from Gen
          end if
        end for
      end for
      $C_k = C_k \cup Gen$
    end if
    $l = l + 1$
  until $Gen = \emptyset$
  $k = k + 1$
  C_{k+1} = candidate_generation($\bigcup_i C_k^i$)
until $C_k = \emptyset$
return HG

HiGens mined between August and October				
Time period	**Generalized itemset**	**Support (%)**	**Gen. level**	**Relationship**
Strongly Stable HiGen 1				
August	{(Service, Twitter), (time, from 9 p.m. to 10 p.m.)}	0.051	2	$\rightsquigarrow$
September	{(Service, Twitter), (time, from 9 p.m. to 10 p.m.)}	0.020	2	$\rightsquigarrow$
October	{(Service, Twitter), (time, from 9 p.m. to 10 p.m.)}	0.022	2	-
Strongly Stable HiGen 2				
August	{(Service, Twitter), (time, from 7 p.m. to 13 p.m.)}	0.071	3	$\rightsquigarrow$
September	{(Service, Twitter), (time, from 7 p.m. to 13 p.m.)}	0.031	3	$\rightsquigarrow$
October	{(Service, Twitter), (time, from 7 p.m. to 13 p.m.)}	0.044	3	-
Weakly Stable HiGen 3				
August	{(User, Employee), (time, from 7 p.m. to 13 p.m.)}	0.051	3	$\rightsquigarrow$
September	{(User, Employee), (time, from 7 p.m. to 13 p.m.)}	0.062	3	$\rightsquigarrow$
October	{(User, John), (time, from 7 p.m. to 13 p.m.)}	0.024	3	-
Monotonous HiGen 4				
August	{(Service, Home_W), (User, John)}	0.005	1	$\nearrow$
September	{(Service, Home_Services), (User, John)}	0.017	2	$\rightsquigarrow$
October	{(Service, Home_Services), (User, John)}	0.021	2	-
Monotonous HiGen 5				
August	{(Service, Travels), (User, John)}	0.009	2	$\searrow$
September	{(Service, Trains), (User, John)}	0.008	1	$\rightsquigarrow$
October	{(Service, Trains), (User, John)}	0.007	1	-
Oscillatory HiGen 6				
August	{(Service, Travels), (User, Terry)}	0.015	2	$\searrow$
September	{(Service, Maps), (User, Terry)}	0.053	1	$\nearrow$
October	{(Service, Travels), (User, Terry)}	0.017	2	-
Oscillatory HiGen 7				
August	{(Service, MMS), (Time, from 13 p.m. to 14 p.m.)}	0.004	2	$\nearrow$
September	{(Service, MMS), (Time, from 13 p.m. to 15 p.m.)}	0.005	3	$\searrow$
October	{(Service, MMS), (Time, from 13 p.m. to 14 p.m.)}	0.002	2	-

Table 6.3: Examples of HiGens. *mDesktop* dataset. $min\_sup = 0.001\%$

Service category	**Number of stable** HiGens (%)			**Number of monotonous** HiGens (%)	**Number of oscillatory** HiGens (%)
	Weak	**Strong**	**Total**		
Travels	18	9	27	32	41
Communication	19	33	52	22	26
Entertainment	20	7	27	34	39
Friends	14	12	26	44	30
Home_Services	17	28	45	32	23
Recs	15	14	29	36	35
Teamlife	14	22	36	27	37

Table 6.4: HiGen distribution. *mDesktop* dataset. $min\_sup = 0.001\%$

Chapter 7

Semi-automatic construction of semantic models

The outstanding growth of both context-aware environments and user-generated content coming from social network communities prompted the investigation of new ways to represent domain knowledge and its relationships. Semantic Web tools provide the instruments to significantly enrich the knowledge representation through a wide range of semantics-based models. These models support users in understanding the meaning of a resource and the related domain. Ontologies are examples of fully comprehensive models for describing domain-specific knowledge. Knowledge representation by means of ontology-based models, entails (i) shared agreement of meaning, (ii) term disambiguation, and (iii) domain description through concepts and relationships. The contribution of both advanced data mining algorithms and semantics-based knowledge representation may enhance the knowledge discovery process in a broad range of application contexts, such as social behavior analysis, knowledge discovery from user-generated content, and Web service personalization.

Useful ontologies for a given application domain can be either provided by domain experts or (semi-)automatically inferred from the data of interest. Although the Semantic Web already provides a full technological stack to access semantics-based resources, most of the existing approaches, such as the creation of ontologies in a Web Ontology Language like the OWL [115], still heavily rely on the human intervention. Hence, the machine-driven construction of meaningful ontologies is becoming an increasingly appealing target in several research fields, including information retrieval, data mining, and data summarization. For example, the exponential growth of social medias like blogs and social network services has significantly increased the

need of useful ontologies to efficiently support the analysis of large data volumes [51]. Thus, novel and more efficient approaches to automatically construct useful ontologies tailored to the analyzed data are desirable.

This chapter presents a effective semi-automatic approach to construct ontologies tailored to structured data [32]. Structured datasets are data collections whose content is organized by means of a schema that describes the relevant data features of interest. For instance, a relational dataset schema is characterized by a set of attributes which describes the main data features. Similarly, an XML dataset is characterized by a set of tags (elements). For the sake of simplicity, we will denote the data features belonging to the dataset schema as attributes in the rest of this paper for both relational and XML data. Ontology construction commonly entails a two-level analysis: (i) an intensional data analysis, to represent shared data concepts and their relationships, and (ii) an extensional data analysis, to represent instances of concepts (i.e., the individuals) and their associations. Different approaches can be used to model and represent ontologies. For instance, Description Logic (DL) [13] can be profitably used to represent ontologies. Ontology representation based on Description Logic relies on two main components: (i) the Tbox component, which includes intensional statements about general concepts, and (ii) the Abox component, which includes both extensional statements about the individuals of those concepts and membership statements, which bind instances with their concepts.

To support domain experts in constructing meaningful ontologies, expressed by using the Description Logic and tailored to the input structured data, the DOnG (Dependency-driven ONtology Generator) framework has been presented. It exploits two well-founded database and data mining techniques, i.e., functional dependency discovery and association rule mining. These technique provide valuable correlations hidden in both the data schema and content. Although these correlations have not the semantic expressiveness of formal ontological statements, they may be used, in conjunction with a formal knowledge base representation, i.e., the Description Logic), to effectively support experts in ontology construction. Given a structured dataset, DOnG infers a Schema ONtology Graph, denoted as SOnG, through the discovery of functional dependencies holding in the schema of the analyzed data. The SOnG provides a graph-based representation of some of the hidden correlations among the attributes of the schema. Attributes are considered as the main concepts at the intensional level over which experts may assert meaningful Tbox ontological statements concerning the analyzed data [13]. Since functional dependencies [90] represent tight correlations among sets of attributes, we exploit them to represent Conceptual relation-

ships among attributes (i.e., concepts). A functional dependency, written as $S_1 \Rightarrow S_2$, states that if, in a structured dataset characterized by a set of attributes $\mathcal{T}=\{t_1, \ldots, t_n\}$, two data instances agree on the value of a set of attributes $S_1 \subseteq \mathcal{T}$, then they must agree on the value of a set of attributes $S_2 \subseteq \mathcal{T}$. We argue that a functional dependency models a Conceptual relationship that is worth considering during the Tbox statement assertion. Next, driven by the analyst-validated dependencies, the DOnG framework analyzes the data content to discover the Schema ONtology Instance Graphs, denoted as SOnIG. A SOnIG is a graph composed of data item, i.e., pairs (attribute,value), linked by association rules. Association rules represent relevant but hidden correlations among data at the instance level. Among the set of discovered rules, the analyst may select and exploit most valuable ones to assert Abox ontological statements, i.e., assertions about individuals and their relationships. Individuals are represented by data items and rules suggest possibly relevant relations. Furthermore, the recurrent data items, i.e., the pairs (attribute, value), may suggest to the expert significant membership assertion (e.g., an individual is an instance of a given concept). The pushing of functional dependency constraints into the rule mining step enables the efficient inference of an ontology instance graph at the data item granularity as well as eases domain expert validation by focusing the investigation on the most relevant item recurrences.

The chapter is organized as follows. Section 7.1 formalizes the problem addressed in this paper, Section 7.2 presents the DOnG framework and describes the main features of its building blocks, while Section 7.2.4 presents, as application scenario, the usage of the inferred semantic models to drive the generalized pattern mining process.

7.1 Problem statement

In a given domain, an ontology is a formal knowledge representation that highlights the main concepts and the relationships among those concepts. In this paper, we focus on ontology construction from structured data.

A structured dataset is characterized by a schema (i.e., a set of conceptual features, also called attributes in the following) and a content (i.e., a set of instances). Let $\mathcal{T}=\{t_1, \ldots, t_n\}$ be a set of attributes and $\Omega=\{\Omega_1, \ldots, \Omega_n\}$ be the corresponding attribute domains. An item is a pair $(t_i, value_i)$ which assigns value $value_i \in \Omega_i$ to attribute t_i. A structured dataset D is a collection of instances, where each instance is a set of items.

The process of ontology construction may be effectively supported by two database and data mining techniques, i.e., functional dependency discovery and association rule mining.

Itemsets and association rules describe the co-occurrence of data items in large data collections [2]. Association rules [2] are usually represented as implications $X \rightarrow Y$[1], where X and Y are arbitrary itemsets such that $X \cap Y = \emptyset$.

Different relationships may hold among data attributes. Functional dependencies, formally stated in the following definition, are a notable and largely established kind of attribute relationships.

Definition 27 *Functional dependency. Let D be a structured dataset and $\mathcal{T}$=$\{t_1, t_2, \ldots, t_n\}$ its attributes. Let $S_1 \subseteq \mathcal{T}$ and $S_1 \subseteq \mathcal{T}$ be two subsets of attributes. D satisfies the functional dependency $S_1 \Rightarrow S_2$ if the following condition holds for every pair of instances r_1 and r_2 in R:*

- *If $r_1.S_1 = r_2.S_1$, then $r_1.S_2 = r_2.S_2$*

where $r_i.S_j$ is the set of values assumed by the attributes in S_j for the record r_i.

For example, consider a zoological dataset for biological classification storing all the information (e.g., the class, the discovery date) about all the species belonging to the animal kingdom discovered in the last five centuries. A functional dependency may hold from the *specie* (e.g., lions) to *class* (e.g., mammals) attributes as it holds a is-a relationship among these concepts. In [70], approximate functional dependencies have been also proposed. They represent functional dependencies that almost hold. Their definition is based on the minimum number of records that needs to be removed from the relation D for the dependency $S_1 \Rightarrow S_2$ to hold in D. The number of records that needs to be removed is represented by the error measure (or by the opposite measure, called strength). The error $\epsilon(S_1 \Rightarrow S_2)$ [62] should be defined as in Definition 28.

Definition 28 *Dependency error. Let D be a structured dataset and $\mathcal{T}$=$\{t_1, \ldots, t_n\}$ be its attributes. Let $S_1 \subseteq \mathcal{T}$ and $S_1 \subseteq \mathcal{T}$ be two subsets of attributes*

[1]We denote as $\cdot \rightarrow \cdot$ an association rule and as $\cdot \Rightarrow \cdot$ a functional dependency throughout the paper.

in D. The dependency error $\epsilon(S_1 \Rightarrow S_2)$ of the dependency $S_1 \Rightarrow S_2$, is defined as:

$$\epsilon(S_1 \Rightarrow S_2) = \frac{min\{|R| \mid R \subseteq D \wedge S_1 \Rightarrow S_2 \text{ holds in } D \setminus R\}}{|D|}$$

Corollary 2 *Dependency strength. Let D be a structured dataset and let $S_1 \subseteq \mathcal{T}$ and $S_1 \subseteq \mathcal{T}$ two sets of attributes in D. The dependency strength $st(S_1 \Rightarrow S_2)$ of a dependency $S_1 \Rightarrow S_2$ is defined as:*

$$st(S_1 \Rightarrow S_2) = 1 - \epsilon(S_1 \Rightarrow S_2)$$

The dependency error $\epsilon(S_1 \Rightarrow S_2)$ has a natural interpretation as the fraction of records with exceptions affecting dependency $S_1 \Rightarrow S_2$. The strength measure represents the opposite information with respect to dependency error. When the strength measure is equal to 100% the dependency holds on the whole dataset, i.e., a functional dependency holds.

The definition of approximate functional dependency [62, 70] follows:

Definition 29 *Approximate functional dependency. Let D be a structured dataset and $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ be its attributes. Let $S_1 \subseteq \mathcal{T}$ and $S_2 \subseteq \mathcal{T}$ be two sets of attributes in D. An approximate functional dependency is an implication in the form $S_1 \rightsquigarrow S_2$, such that*

1. $S_1 \subseteq T$ *and* $S_2 \subseteq T$
2. $S_1 \cap S_2 = \emptyset$
3. $st(S_1 \Rightarrow S_2) < 1$

An approximate dependency satisfies a minimum strength threshold *min_st* if its strength is higher than *min_st*. Usually, only functional dependencies and approximate functional dependencies satisfying *min_st* are considered. The dependencies characterized by a dependency strength lower than the threshold are usually not of interest as they do not represent schema correlations holding for the majority of the data.

Dependencies and association rules can be exploited to provide to analysts information useful for semi-automatic ontology construction. Two graph-based representations of the most significant correlations holding among the schema and the content of the analyzed data, i.e., the schema ontology graph and the schema ontology instance graph, are proposed and exploited to support the expert in ontology construction. Their definitions follows.

7.1.1 Schema Ontology Graph

A schema ontology $\mathcal{O}$ is a graph-based representation of the (approximate) dependencies holding in a structured dataset and their strengths. Its definition follows.

Definition 30 **Schema Ontology Graph (SOnG).** *Let D be a structured dataset and $\mathcal{T}=\{t_1, t_2, \ldots, t_n\}$ be its attributes. Let min_st be a minimum dependency strength threshold. A SOnG $\mathcal{O}$ is an oriented graph $\Gamma=\langle \mathcal{T}, E^* \rangle$, whose nodes are $t_i \in \mathcal{T}$. An oriented edges $e_{ik} = (t_i, t_k) \in E^*$ represents a functional dependency or an approximate functional dependency satisfying min_st that holds from t_i to t_k and is labeled with a weight $w_{ik}=st(t_i \Rightarrow t_k)$.*

Graph nodes are attributes while edges represent dependencies, eventually approximate, holding among couples of attributes. Edge weights take values in the range (0,1] and represent the attribute dependency strength, i.e., the fraction of instances for which each dependency holds. Figure 7.1 shows an example of a simplified SOnG obtained by enforcing a minimum dependency strength equal to 95%, i.e., only the edges that correspond to dependencies characterized by a dependency strength higher than 95% are included in the SOnG reported in Figure 7.1. It concerns a portion of the (simplified) DBLP XML dataset [75], which collects information about conference proceedings. For instance, the concept *Author name* may be meaningfully associated with the author *Affiliation* (each authors is related to an affiliation, or a set of affiliations in some cases).

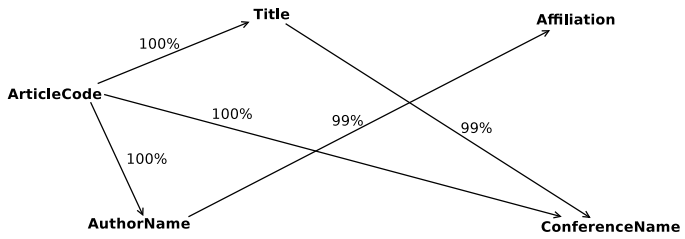


Figure 7.1: A simplified SOnG built over the example DBLP XML attributes.

7.1.2 Schema Ontology Instance Graph

Given a SOnG built over the data schema, the corresponding graph at the data item level could be defined. A graph-based representation, namely the SOnIG (Schema Ontology Instance Graph), of the correlations holding among couples of items, belonging to the structured dataset D, such that their corresponding attributes are (approximately) functionally dependent in D is proposed in the following.

Definition 31 Schema Ontology Instance Graph (SOnIG). *Let $\mathcal{T}=\{t_1, t_2, \ldots, t_n\}$ be the attributes which describe a structured dataset D and let $\mathcal{I}$ be the set of all data items in D. Let $\mathcal{O}=\langle \mathcal{T}, E^* \rangle$ be a SOnG built over D. Let $\Phi : E^* \rightarrow \mathcal{R}$ be an application that maps each oriented edge $e_{ik} = (t_i, t_j) \in E^*$ to the set R_{ij} of 2-length association rules (t_i,$value_i$) $\rightarrow$ (t_j,$value_j$) holding from items belonging to attribute t_i to items belonging to attribute t_j. The schema ontology instance graph is an oriented graph $\Theta=\langle \mathcal{I}, \Phi(E^*) \rangle$ whose nodes are distinct data items in D and the oriented edges represent association rules holding between couples of items. An oriented edge from item (t_i,$value_i$) to item (t_j,$value_j$) is weighted by the confidence of the corresponding association rule (t_i,$value_i$) $\rightarrow$ (t_j,$value_j$).*

Starting from the structured dataset D and a SOnG Γ (Cf. Definition 30), the SOnIG is constructed by representing data items in D and the association rules involving couples of items belonging to (approximately) functionally dependent attributes. The SOnIG edge weight, i.e., the rule confidence, represents the strength of the correlations among data items. To extract the SOnIG we exploit the association rule mining process [2] without enforcing any mining constraint (i.e., absolute minimum support threshold = 1 and minimum confidence = 0%). Since it is focused on association rules between couples of items we only mine 2-length rules.

Figure 7.2 shows a portion of SOnIG built over the example DBLP XML dataset (see Figure 7.4). The edge between {*(Author name, Rossi)*} and {*(Affiliation, Politecnico di Torino)*} is weighted by 78% because the confidence of the rule {*(Author name, Rossi)*} $\rightarrow$ {*(Affiliation, Politecnico di Torino)*} is 78%. Notice that, according to Definition 31, an oriented edge between two data items is omitted if the attributes of the corresponding items are not dependent, e.g., the edge directed from {*(Author name, Rossi)*} to *(Title, A performance evaluation for ...)*.

In this following section, a framework, called DOnG, that extracts both SOnGs (Cf. Definition 30) and SOnIGs (Cf. Definition 31) from structured

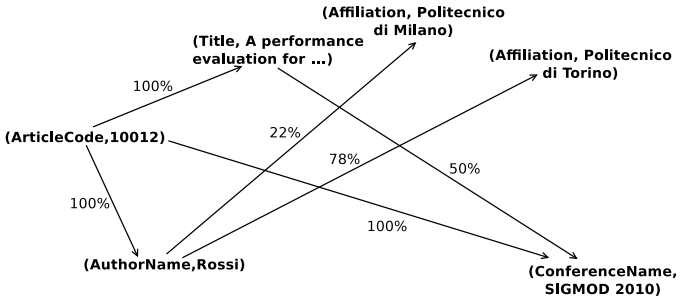


Figure 7.2: An example of the SOnIG over some DBLP XML records enriched with confidence values.

datasets to support analysts in the construction of meaningful ontologies is presented.

7.2 The dependency-driven ontology generator

This section describes the DOnG (Dependency-driven Ontology Generator) framework, which entails ontology construction from structured data supported by a two-step mining process, which entails: (i) approximate functional dependency discovery, and (ii) association rule mining.

Figure 7.3 shows the building blocks of the DOnG framework, which addresses the semi-automatic construction of a meaningful ontology, expressed by the Description Logic, tailored to the input data. Description Logics [13] formalize the ontology representation in terms of a set of shared concepts and their relationships. It includes two distinct parts: (i) the Tbox, which includes a finite set of assertions about shared primitive concepts, inferred from the data of interest, and the binary relations among them, and (ii) the Abox, which includes membership assertions (e.g., an individual is an instance of a given concept) as well as extensional statements about the concept instances (belonging to the analyzed data) and their relationships.

In DOnG, the Tbox and Abox statement assertion steps are fully supported by an ad-hoc mining process. The mining process makes use of a two-step analysis: (i) functional dependency discovery from the data schema (i.e., the attribute level), and (ii) association rule mining from the data in-

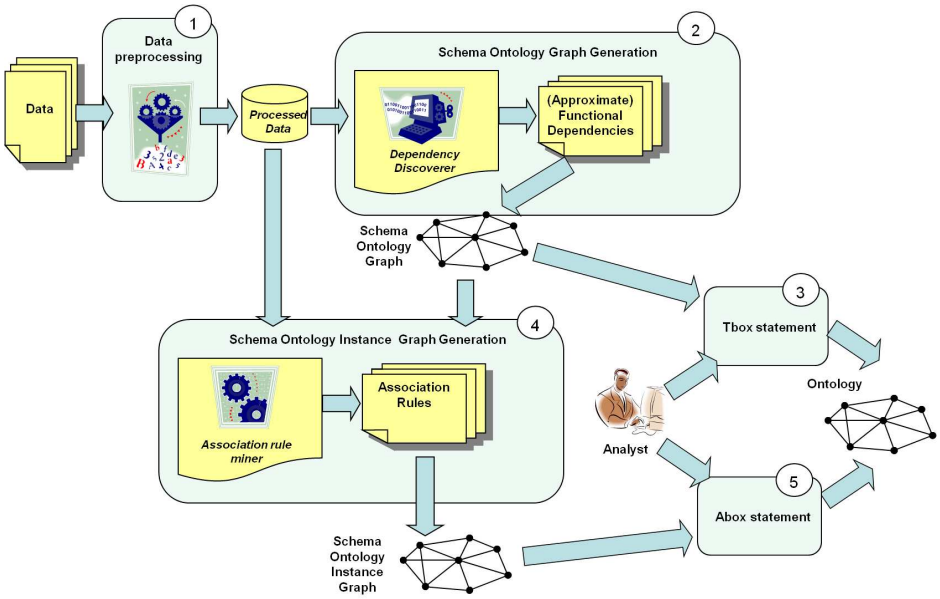


Figure 7.3: The DOnG framework architecture.

stances (i.e., the item level). The first step of the mining process supports the construction of the Tbox part of the ontology representation, while the second one supports the Abox construction step. Functional dependencies and approximate functional dependencies are discovered and represented in a graph-based model, i.e., the SOnG (Cf. Definition 30). The generated model is validated by the analyst and it is used to semi-automatically generate the Tbox component of the description logic based ontology representation. In particular, the analyst selects the dependencies that are worth considering in the Tbox construction phase. Validated dependencies are then used to drive the rule mining process at the item level. This enables the generation of a reduced amount of relevant rules, thus, easing the analyst validation of the extracted recurrences. Association rules and the related data items are represented in a graph-based model, i.e., SOnIGs (Cf. Definition 31), that effectively support the analysts during the Abox generation phase. In particular, each rule is a potential statement in the defined Abox.

7.2.1 Data preprocessing

The DOnG addresses ontology construction from structured data. A structured dataset is a dataset characterized by a schema, which describes its main data features. Most common examples of structured data are the relational and the XML data. A relational dataset is described by a set of attributes, while an XML dataset is organized by a set of tags. To enable both functional dependency discovery and association rule mining from generic structured data, the DOnG framework requires a preprocessing step to make data suitable for the mining process. Since the application of the aforementioned data mining techniques to relational data is straightforward [2, 62], in this section we mainly focus on describing how to make XML data suitable for both functional dependency and association rule discovery.

Most of the well-founded models proposed in the past (see, for example, OEM [91], UnQL [29], and GraphLog [38]) lend themselves to represent XML data by graphs whose nodes denote either objects (i.e., abstract entities) or values (i.e., primitive values), while edges represent relationships between them. According to [40], we represent an XML document as a labeled tree[2], as stated in Definition 32:

Definition 32 *An XML document is a labeled tree $\langle N, E, r\rangle$, such that:*

- *N is the set of nodes;*
- *$r \in N$ is the root of the tree (i.e., the root of the XML document);*
- *E is the set of undirected edges;*

Figure 7.4 reports a portion of the DBLP XML document represented as a tree-based structure (with the considered labels for nodes and edges). It will be used as running example in the rest of the paper. The XML document reports information about a number of conference proceedings. Each article is characterized by its code, title, authors' name and affiliation, acceptance date, conference name, and location. Attributes and elements are characterized by empty circles, whereas the textual content of the elements and the attribute values are reported as black-filled circles. The node labeled

[2]Note that XML documents are here tree-like structures (and not generic graphs) because, following the so-called literal semantics, we do not interpret referencing attributes as pointers.

as {} is the root of the XML document. The labeled tree rooted in the *Article* node is an example of conference proceeding. In general, the elements may be connected to either their sub-elements, attributes, or leaf node, while the attribute nodes may be connected to leaf nodes that represent their corresponding attribute values.

Moreover, the following two properties regarding graph nodes and edges hold: (1) Each node n_i has a tuple of labels $NL_i = \langle Ntag_i, Ntype_i, Ncontent_i \rangle$. The label type $Ntype_i$ indicates whether the node is a root, an element, a text, or an attribute, whereas the label $Ncontent_i$ can take as value a PCDATA [3] or $\perp$ (undefined, for nonterminals).

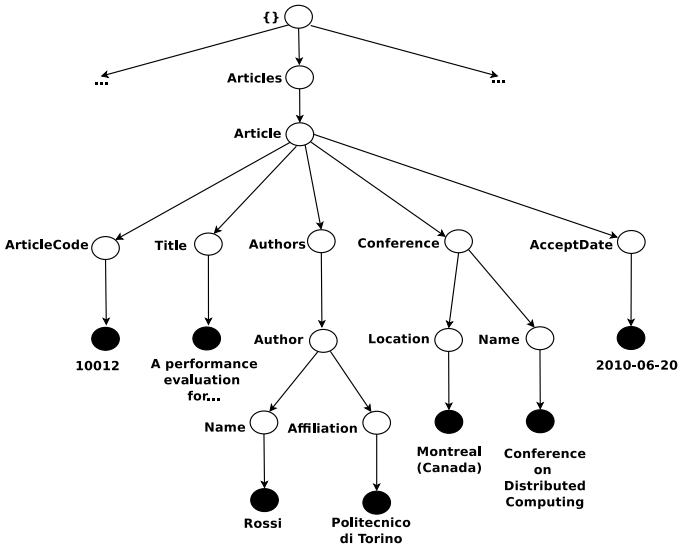


Figure 7.4: A simplified labeled tree of a portion of the example DBLP XML document

Since the focus is to find correlations among elementary values of the XML documents, and such values may occur as either textual content of leaf elements or attribute values, I only consider a single node type and I do not specify any type label. In the following, the notion of item in the XML data format is defined.

Definition 33 *XML document item. Let* $\langle N, E, r \rangle$ *be an XML document and* $r_r \in N$ *be the root of the corresponding document. An item is a couple*

[3]PCDATA is a reserved keyword used in the DTD XML Document Type Definition (DTD) which stands for Parsed Character Data and indicates a generic plain text.

(attribute, value) that could be represented in the XML document $\langle N, E, r\rangle$ *as a path connecting a direct descendant* $n_{dr} \in N$ *of the root* r_r *to a leaf, i.e., a terminal node* $n_n \in N$*. The path corresponds to an item such that:*

1. *attribute is the label of the element with a content rooted in* n_{dr} *and defined as a sub-path from* n_{dr} *to the element.*
2. *value is the content of the considered element.*

Consider again the example DBLP XML dataset reported in Figure 7.4. Suppose to set the node labeled as *Articles* as root of the dataset instances. An example of item is the couple (*Article Title*, *A performance evaluation of ...*) that identifies a sub-path connecting node labeled as *Article* to the element node labeled as *Title*.

XML documents can be analyzed by different data mining techniques, such as association rule extraction [28], taxonomy inference [36], and data summarization [20], to discover interesting recurrences among data items. To exploit well-known techniques proposed for relational datasets, the XML document is usually transformed in a relational dataset as stated by the following definition.

Definition 34 *XML in the relation data format. Let* $\langle N, E, r\rangle$ *be an XML document and* $r_r \in N$ *be a root of the corresponding document. Let* $T = \{t_1, t_2, \cdots, t_n\}$ *be the set of attributes. A relational dataset is a collection of instances, where each instance* r *is a set of items* $\{(t_1, value_1), \cdots, (t_n, value_n)\}$ *in* $\langle N, E, r\rangle$*.*

To transform an XML document into the corresponding relational format, an approach similar to that proposed in [20] is adopted. However, a relational representation is generated, while the authors of [20] transform each XML document into a transactional dataset. This generation is described in the following. It first selects as instance root n_{dr} the appropriate node in the labeled tree representing the XML document. Items are couples of $(t_i, value_i)$, where the attribute t_i is the label of an element with a content rooted in the instance root and defined as a complete sub-path from the root to the element, while $value_i$ is the content of the considered element. Each subtree, rooted in n_{dr}, defines an instance in the XML dataset.

Consider again the example DBLP XML dataset. The corresponding relational dataset is a set of per-author article records, each one composed of

a set of items describing, for a specific author, a published proceeding. The corresponding relation achieved by tailoring the XML data to the relational data format is the following

DBLP(**Article code**, **Author name**, `Title`, `Conference name`, `Conference location`, `Author affiliation`)

in which the pair (*Article code*,*Author name*) represents the primary key of the relation.

The $(t_i, value_i)$ representation for data items yields a single item when the XML element contains a single data value and when it has a textual content. Even in the case in which the textual content can be considered as a multiple value element (e.g., the article author may assume multiple values), data preprocessing yields a single data value to preserve the relational data schema (Cf. Definition 34). The presence of multiple value elements may produce a set of records each one involving distinct single item values.

To reduce side effects due to multiple values and textual data noise, textual elements may be preprocessed by performing stopword elimination (i.e., elimination of very frequent and noisy words such as articles) and stemming (i.e., reduction of words to their semantic stem) [92, 93]. Both operations are commonly performed in textual data management.

7.2.2 Schema Ontology Graph generation

The aim of this block is to build a SOnG based on the schema of the input structured dataset to effectively support the analyst in the Tbox ontological statement assertion. Since the challenge is to figure out shared data concepts and their relationships data attributes belonging to the schema as potentially relevant concepts and the notion of approximate functional dependencies [90] (Cf. Definition 29) are exploited to highlight tight correlations among attributes.

The SOnG generation block of the DOnG framework takes in input both the preprocessed structured dataset and a minimum dependency strength (Cf. Definition 2), and generates a schema ontology graph (Cf. Definition 30) built over the schema of the analyzed data. To this aim, three main steps are performed:

- *Functional dependency and approximate functional dependency discov-*

ery. To discover functional and approximate functional dependencies it exploits the traditional TANE algorithm [62], which is a level-wise algorithm in which, at the k-th iteration, dependencies involving attribute sets of size at most equal to k are considered. A brief overview of the adopted approach [62] is reported in the following. Nevertheless, the proposed DOnG framework enables users to easily integrate diverse algorithms as well.

- *Analyst validation.* The analyst validates the inferred dependencies and selects the most relevant ones for the ontology Tbox construction process. The domain expert is in charge of discriminating among relevant data attributes and functional dependencies holding among them and, eventually, pruning uninteresting ones.

- SOnG *building.* This step generates a SOnG (Schema Ontology Graph) according to Definition 30. It includes each validated dependency as an edge connecting two data attributes (i.e., the SOnG nodes). Nodes are the selected attributes while the oriented edges represent dependencies among schema concepts that are worth considering from the analyst's point of view. An edge connecting two arbitrary nodes is weighted by the strength of the approximate functional dependency between the corresponding nodes. The higher is the dependency strength (Cf. Definition 2) the higher is the correlation strength in the source dataset.

Consider again the previous example, and suppose that, by enforcing a minimum dependency strength threshold equal to 80%, the set of approximate dependencies reported in Table 7.1 is discovered from the example dataset. Notice that a dependency strength equal to 100% implies that the approximate dependencies always holds in the source dataset (i.e., it is a functional dependency).

Table 7.1: Discovered dependencies from a portion of DBLP XML dataset.

Num	**Dependency**	**strength (%)**
1	*Article code* $\rightsquigarrow$ *Title*	100
2	*Article code* $\rightsquigarrow$ *Conference name*	100
3	*Article code* $\rightsquigarrow$ *Author name*	100
4	*Author name* $\rightsquigarrow$ *Author affiliation*	99
5	*Title* $\rightsquigarrow$ *Conference name*	99
6	*Title* $\rightsquigarrow$ *Author affiliation*	90

Consider now the last approximate dependency *Title* $\rightsquigarrow$ *Author affiliation* with strength equal to 90% reported in Table 7.1. It highlights a correlation

that might be of minor interest from the domain expert point of view. Expert pruning may discard the uninteresting dependency *Title* $\rightsquigarrow$ *Author affiliation* discovered from D. Since an early dependency filtering based on minimum dependency strength threshold min_st has been enforced, the analyst may also vary the mining constraint (e.g., by setting a higher minimum strength threshold) to tailor the discovered dependency set to her/his actual needs. An experimental analysis on the impact of the maximum dependency strength threshold is reported in Section *Experimental results*.

The schema ontology graph obtained after the expert validation of the approximate dependencies in Table 7.1 is reported in Figure 7.1.

The TANE algorithm [62], whose pseudo code is reported in Algorithm 5, searches for approximate dependencies that (i) satisfy the minimum strength threshold min_st, and (ii) are minimal (i.e., the right-hand side attribute set is not functional dependent with any subset of the left-hand side set). TANE is a level-wise algorithm in which, at the k-th iteration, dependencies having the left-hand size in L_{k-1} are computed (line 6). A level L_k is a collection of attribute sets of size k. Then, the minimality property and the minimum strength threshold are exploited to prune L_{k-1} and, thus, avoid exhaustive lattice exploring. Finally, level L_k is generated starting from L_{k-1} (line 8).

To discover functional dependencies that involve just single attributes at both the left-hand and the right-hand dependency side, the iterative procedure described above is stopped when iteration $k = 2$ is completed. Thus, the redundant knowledge extraction is prevented.

Algorithm 5 The TANE algorithm.

Input: relation D over schema $\mathcal{R}$, minimum strength threshold min_st
Output: minimum not trivial approximate functional dependencies that hold in D
1: $L_0 = \emptyset$
2: $C^+(\emptyset) = D$
3: $L_1 = \{\{A\} | A \in R\}$
4: $k = 1$
5: **while** $L_k \neq \emptyset$ **do**
6: compute_Dependencies(L_k)
7: prune(L_k, min_st)
8: L_{k+1}= generate_Next_Level(L_k)
9: $k = k + 1$
10: **end while**

Algorithm 6 Schema ONtology Instance Graph extraction.

Input: SONG Γ, structured dataset D
Output: SONIG Θ
1: L_1 = set of items in D
2: L_2 = itemset_generation(L_1, Γ) /*/* *2-itemsets generation* */*/
3: scan D and count support for each $c \in L_2$
4: *Rules* = rule_generation(L_1 , L_2)
5: **for all** r in *Rules* **do**
6: **if** expertvalidation(r) = 'discard rule' **then**
7: remove r from *Rules*
8: **end if**
9: **end for**
10: Θ = buildSchemaOntologyInstanceGraph(*Rules*, Γ)
11: **return** L

7.2.3 Schema ONtology Instance Graph generation

The last step of the DONG framework entails the generation of the SONIG (Schema ONtology Instance Graph) built over the analyzed data (Cf. Definition 31), which might be profitably exploited to drive the generalized pattern mining and knowledge discovery process (see Section 7.2.4). To this aim, it takes in input the preprocessed structured dataset D, and the SONG (Schema ONtology Graph) Γ. It produces a SONIG (Cf. Definition 31) by performing 2-length association rule mining driven by the previously extracted and validated schema dependencies.

Algorithm 6 reports the pseudo-code of the SONIG mining step of the DONG framework. It adopts an Apriori-like approach to frequent itemset mining [2]. The traditional Apriori-like miner iteratively generates frequent itemsets by following a level-wise approach. First, it extracts itemsets of length k and then itemsets of length $k+1$ by combining frequent k-itemsets. Finally, association rules are mined from the set of frequent itemsets.

Since exclusively the rules that involve couples of items are needed to generate the SONIG, and in particular only those rules that are instances of the SONG (Cf. Definition 31), Algorithm 6 extracts only items (line 1) and 2-itemsets (lines 2-3). Hence, differently from traditional Apriori-like algorithms [2], a constraint on the maximum length of the mined itemsets is enforced. By exploiting 1-itemsets and 2-itemsets, only those 2-length rules that are instances of SONG are mined (line 4).

To check their semantic soundness, the analyst may look into the content of the extracted rules and, possibly, discard uninteresting ones (lines 5-9).

For instance, suppose now that both rules X={*(Article code, 10012)*} $\rightarrow$ {*(Title, A performance evaluation for ...)*} and Z={*(Author name, Rossi)*} $\rightarrow$ {*(Author affiliation, Politecnico di Torino)*} are frequent and their cor-

responding dependencies have been validated by the analyst. Since they are both instances of the SOnG (see Figure 7.1), they are extracted and validated by the analyst.

After the expert validation step, only validated rules are included into *Rules*. Selected rules are exploited to represent the output SOnIG Θ related to SOnG Γ and the source dataset D. The SOnIG building phase (line 10) generates a valid SOnIG, according to Definition 31. It includes each validated rule as an instance of a relationship (i.e., an oriented edge) among data items belonging to the rules (i.e., SOnG nodes). Consider again rule X={*(Article code, 10012)*} $\rightarrow$ {*(Title, A performance evaluation for ...)*}. By looking into the SOnG reported in Figure 7.1, the attribute node *Article code* is connected to node *Title* through an oriented edge. Thus, the corresponding edge is included into the SOnIG as reported in Figure 7.2, and represents the rule X.

The generated SOnIG is used by the expert during the Abox ontology construction. More specifically, data items belonging to the SOnIG may suggest to the expert possibly relevant membership statements (e.g., 10012 *is an instance of the article code*) while validated association rules may prompt the assertion of extensional statements (e.g., *the article code* 10012 *is associated with the article title A performance evaluation for ...*).

7.2.4 Application scenario: generalized pattern mining

An ontology is a knowledge representation that models the main concepts and their relationships.

In the following, we propose an application scenario in which the DOnG framework may be profitably exploited in a business-oriented environment. A data mining algorithm [16] that heavily relies on a user-provided domain knowledge representation has been presented in Chapter 4. It addresses generalized itemset mining [104], which extends the traditional frequent itemset mining proposed in [2] by exploiting a high level abstraction of the mined patterns to prevent relevant knowledge pruning. By evaluating a hierarchical set of aggregations built over data items (i.e., a tree-based schema ontology instance graph), they can be aggregated based on different granularity concepts (generalized items). Each generalized itemset, which is a high level representation of a set of lower level itemsets, can be used both to (i) give a higher level view of patterns hidden in the analyzed data and (ii) represent

knowledge pruned by discarding infrequent lower level itemsets.

However, domain experts are typically asked to develop reliable hierarchical Schema ONtology Instance Graphs over data items from scratch. Manual generation of ad-hoc aggregation hierarchies is often subject to errors and inconsistencies due to (i) data sparsity, (ii) limited analyst experience, or (iii) limited time available for context learning and data processing. Indeed, a machine-driven generation of meaningful hierarchies of data item aggregations is desirable.

A tree-based organization of Schema Ontology Instances extracted by the DONG framework could be effectively exploited to drive the generalization process over data items. Consider the following 3-itemset extracted from the TPC-H Part XML dataset [111]:

$$X=\{(Brand, 13), (Container, SM\_PKG), (Size, 1)\} \text{ (sup=0.03\%)}$$

It provides relevant business-oriented information concerning transport of products of a specific brand. However, mining such pattern may require enforcing a very low support threshold. This task may become unfeasible or could lead to the extraction of an unmanageable amount of patterns. Suppose to enforce a minimum support threshold min_sup=0.05%. Since the above pattern is infrequent, it is not extracted. By considering the exact functional dependency $Brand \rightsquigarrow Manufacturer$ extracted from the TPC-H Part dataset [111] the DONG framework may generate the corresponding schema ontology instance graph over data items, which, in turn, can be exploited to drive the generalization process over item $(Brand, 13)$. The soundness of generated SONIG is validated by a domain expert during the validation step. The generalization process over itemset X produces the following generalized patterns Y:

$$Y=\{(ManufacturerID, 1), (Container, SM\_PKG), (Size, 1)\}$$
$$\text{(sup=0.08\%)}$$

The brand is generalized as its corresponding manufacturer. Since the pattern is a higher level aggregation of knowledge represented in itemset X, it is characterized by an higher support. Note that the generalized itemset becomes frequent with respect to the minimum support threshold.

The machine-driven generation of a tree-based SONIG may effectively drive generalized itemset mining. The DONG framework may produce a semantics-based data representation suitable for driving the knowledge discovery process in different application contexts.

Chapter 8

Conclusions

The effectiveness of the data mining and knowledge discovery (KDD) process strictly depends on the granularity level of the analyzed data. The availability of semantic models built over the source data, which represent the most notable aggregations and relationships, may significantly enhance both the performance of the mining phase and the usefulness of the generated pattern-based models to drive the knowledge discovery process.

This work thoroughly described the process of itemset and association rule mining in the presence of taxonomies, which entails the discovery of the most significant correlations hidden in the analyzed data at different abstraction levels. It presented an algorithm, namely GenIO (GENeralized Itemset DiscOverer) that performs an opportunistic aggregation of the knowledge occurring in the source data to prevent the extraction of redundant higher level correlations. The effectiveness of the presented algorithm in supporting knowledge discovery in real application contexts, i.e., context-aware user and service profiling, network traffic analysis, and social network analysis, has been also analyzed and validated by domain experts.

The suitability of the extracted correlations for being used in the decision making process also depends on their cardinality and characteristics of the mined patterns. This work has analyzed the use of constraints to drive the generalized pattern mining process as well. Constraints, enforced both during the mining phase and as a postprocessing step, allow early pruning not interesting higher level correlations and making the generated pattern sets manageable by domain experts.

Pattern evolution over time may reflect most significant temporal recurrences. The analysis of the evolution of the patterns discovered in consecutive

time periods in the presence of taxonomies has been also addressed by this work. In particular, pattern generalization is profitably exploited to keep track of the most significant variations of the analyzed knowledge without discarding rare but possibly relevant patterns.

Finally, the problem of semi-automatically discovering semantic models used to drive the generalized pattern mining process has been addressed by exploiting both dependency discovery techniques and association rule mining algorithms. The presented techniques are shown to be suitable for driving the construction of either simple hierarchical models, e.g., taxonomies, or even more complex domain knowledge representations, e.g., the ontologies.

Currently, a significant research effort has been devoted to combining the efficiency and the effectiveness of pattern discovery techniques with the usefulness of semantic model inference and analysis. In the next years, the increasing availability of semantic data will even more focus the attention of the Data Mining and the Semantic Web research communities on the use of semantics-based models to drive the data mining and knowledge discovery process. In this reasearch context, the discovery and application of generalized patterns represent a promising research direction.

List of Figures

List of Tables

Bibliography

[1] S. Abrol and L. Khan. Twinner: understanding news queries with geo-content using twitter. In *In Proceedings of the 6th Workshop on Geographic Information Retrieval.*, pages 1–8, 2008.

[2] R. Agrawal, T. Imielinski, and Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD 1993*, pages 207–216, 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th VLDB conference*, pages 487–499, 1994.

[4] R. Agrawal and R. Srikant. Mining quantitative association rules in large relational tables. *ACM SIGMOD International conference on management of data*, pages 1–12, 1996.

[5] R. Agrawal and R. Srikant. Mining association rules with item constraints. In *KDD 1997*, pages 67–73, 1997.

[6] Rakesh Agrawal and Giuseppe Psaila. Active data mining, 1995.

[7] M. L. Antonie, O. R. Zaiane, and A. Coman. Application of data mining techniques for medical image classification. *MDM/KDD'01*, 2001.

[8] D. Apiletti, E. Baralis, T. Cerquitelli, and V. D'Elia. Characterizing network traffic by means of the netmine framework. *Computer Networks 53(6)*, 53(6):774–789, 2009.

[9] Annalisa Appice, Margherita Berardi, Michelangelo Ceci, and Donato Malerba. Mining and filtering multi-level spatial association rules with ares. In Mohand-Said Hacid, Neil V. Murray, Zbigniew W. Ras, and Shusaku Tsumoto, editors, *Foundations of Intelligent Systems, 15th International Symposium*, pages 342–353, Berlin/Heidelberg, Germany, 2005. Springer.

[10] Wai-Ho Au and Keith C. C. Chan. Mining changes in association rules: a fuzzy approach. *Fuzzy Sets Syst.*, 149:87–104, January 2005.

[11] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. *The Semantic Web*, pages 722–735, 2007.

[12] Siyamand Ayubi, Maybin K. Muyeba, Ahmad Baraani, and John Keane. An algorithm to mine general association rules from tabular data. *Information Sciences*, 179(20):3520 – 3539, 2009.

[13] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[14] Lee Bae-Hee, Kim Heung-Nam, Jung Jin-Guk, and Jo Geun-Sik. Location-based service with context data for a restaurant recommendation. *DEXA 2006*, pages 430–438, 2006.

[15] M. Baldi, E. Baralis, and F. Risso. Data mining techniques for effective and scalable traffic analysis. *International Symposium on Integrated Network Management*, pages 105–118, 2005.

[16] E. Baralis, L. Cagliero, T. Cerquitelli, V. D'Elia, and P. Garza. Support driven opportunistic aggregation for generalized itemset extraction. In *5th IEEE International Conference on Intelligent Systems*, pages 102–107, 2010.

[17] Elena Baralis, Luca Cagliero, Tania Cerquitelli, and Paolo Garza. Generalized association rule mining with constraints. *Information Sciences*, In press.

[18] Elena Baralis, Luca Cagliero, Tania Cerquitelli, Paolo Garza, and Marco Marchetti. Cas-mine: Providing personalized services in context-aware applications by means of generalized rules. *Knowl. Inf. Syst.*, 2010.

[19] Elena Baralis, Luca Cagliero, Tania Cerquitelli, Paolo Garza, and Marco Marchetti. Cas-mine: providing personalized services in context-aware applications by means of generalized rules. *Knowl. Inf. Syst.*, 28(2):283–310, 2011.

[20] Elena Baralis, Paolo Garza, Elisa Quintarelli, and Letizia Tanca. Answering xml queries by means of data summaries. *ACM Trans. Inf. Syst.*, 25(3), 2007.

[21] Steffan Baron, Myra Spiliopoulou, and Oliver GÃ$\frac{1}{4}$nther. Efficient monitoring of patterns in data mining environments. In Leonid Kalinichenko, Rainer Manthey, Bernhard Thalheim, and Uwe Wloka, editors, *Advances in Databases and Information Systems*, volume 2798

of *Lecture Notes in Computer Science*, pages 253–265. Springer Berlin Heidelberg, 2003.

[22] P. Basile, D. Gendarmi, F. Lanubile, and G. Semeraro. Recommending smart tags in a social bookmarking system. volume 2, pages 22–29, 2007.

[23] Roberto J. Bayardo, Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. *Data Min. Knowl. Discov.*, 4(2/3):217–240, 2000.

[24] M. Bender, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J.X. Parreira, R. Schenkel, and G. Weikum. Exploiting social relations for query expansion and result ranking. In *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, pages 501 –506, 2008.

[25] C. Bolchini, C. A. Curino, E. Quintarelli, F.A. Schreiber, and L. Tanca. A data-oriented survey of context models. *ACM SIGMOD Record*, 36(4):19–26, 2007.

[26] Mirko Böttcher, Detlef Nauck, Dymitr Ruta, and Martin Spott. Towards a framework for change detection in data sets. In Max Bramer, Frans Coenen, and Andrew Tuson, editors, *Research and Development in Intelligent Systems XXIII*, pages 115–128. Springer London, 2007.

[27] Nicholas A. Bradley and Mark D. Dunlop. Toward a multidisciplinary model of context to support context-aware computing. *Hum.-Comput. Interact.*, 20(4):403–446, 2005.

[28] Daniele Braga, Alessandro Campi, and Stefano Ceri. XQBE (XQuery By Example): A visual interface to the standard XML query language. *ACM Transaction on Database Systems*, 30(2):398–443, 2005.

[29] Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan Suciu. A Query Language and Optimization Techniques for Unstructured Data. In *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 505–516, New York, NY, USA, 1996. ACM Press.

[30] Hee Byun and Keith Cheverst. Utilizing context history to provide dynamic adaptations. *Applied Artificial Intelligence*, 18(6):533–548, 2004.

[31] Luca Cagliero. Discovering temporal change patterns in the presence of taxonomies. *IEEE Trans. Knowl. Data Eng.*, In press.

[32] Luca Cagliero, Tania Cerquitelli, and Paolo Garza. Semi-automatic ontology construction by exploiting functional dependencies and association rules. *Int. J. Semantic Web Inf. Syst.*, 7(2):1–22, 2011.

[33] Luca Cagliero and Alessandro Fiori. Analyzing twitter user behaviors and topic trends by exploiting dynamic rules. In Longbing Cao and Yu Philip, editors, *Behavior Computing: Modeling, Analysis, Mining and Decision*. Springer Berlin Heidelberg.

[34] Luca Cagliero and Alessandro Fiori. Twecom: topic and context mining from twitter. In Longbing Cao and Yu Philip, editors, *The influence of Technology on Social Network Analysis and Mining*. Springer LNSN.

[35] David Wai-Lok Cheung, Jiawei Han, Vincent Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of the Twelfth International Conference on Data Engineering*, ICDE '96, pages 106–114, Washington, DC, USA, 1996. IEEE Computer Society.

[36] Chris Clifton, Robert Cooley, and Jason Rennie. Topcat: Data mining for topic identification in a text corpus. In *In Proceedings of the 3rd European Conference of Principles and Practice of Knowledge Discovery in Databases*, 2002.

[37] G Cong, A K H Tung, X Xu, F Pan, and J Yang. Farmer: finding interesting rule groups in microarray datasets. *ACM SIGMOD 04, Paris, France*, 2004.

[38] Mariano P. Consens and Alberto O. Mendelzon. Graphlog: a Visual Formalism for Real Life Recursion. In *PODS '90: Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 404–416, New York, NY, USA, 1990. ACM Press.

[39] Apiletti D., Baralis E., Cerquitelli T., and D'Elia V. Network digest analysis by means of association rules. In *International IEEE Conference on Intelligent Systems*, 2008.

[40] E. Damiani, B. Oliboni, E. Quintarelli, and L. Tanca. Modeling Semistructured Data by using graph-based constraints. Technical Report 27/03, Politecnico di Milano. Dipartimento di Elettronica e Informazione, July 2003.

[41] Mariam Daoud, Lynda Tamine-Lechani, and Mohand Boughanem. Towards a graph-based user profile modeling for a session-based personalized search. *Knowl. Inf. Syst.*, 21(3):365–398, 2009.

[42] Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. In *Machine Learning*, pages 143–175, 2000.

[43] Guozhu Dong, Jiawei Han, Joyce M. W. Lam, Jian Pei, Ke Wang, and Wei Zou. Mining constrained gradients in large databases. *IEEE Trans. on Knowl. and Data Eng.*, 16:922–938, August 2004.

[44] Guozhu Dong and Jinyan Li. Mining border descriptions of emerging patterns from dataset pairs. *Knowl. Inf. Syst.*, 8:178–202, August 2005.

[45] E. Eskin, L. Portnoy, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *ACM CSS Workshop on Data Mining Applied to Security*, 2001.

[46] Le F., Lee S., Wong T., Kim H.S., and Newcomb D. Minerals: using data mining to detect router misconfigurations. In *MineNet '06*, pages 293–298, 2006.

[47] M. Faloutsos, T. Karagiannis, and K. Papagiannaki. Blinc: multilevel traffic classification in the dark. In *SIGCOMM '05*, pages 229–240, 2005.

[48] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029, 1993.

[49] Stephen C. Gates, Wilfried Teiken, and Keh-Shin F. Cheng. Taxonomies by the numbers: building high-performance taxonomies. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 568–577, New York, NY, USA, 2005. ACM.

[50] Jonna Hakkila and Jani Mantyjarvi. Collaboration in context-aware mobile phone applications. *Hawaii International Conference on System Sciences*, 1:33a, 2005.

[51] M. Hamasaki, Y. Matsuo, T. Nishimura, and H. Takeda. Ontology extraction by collaborative tagging with social networking. 2008.

[52] J. Han and Y. Fu. Mining multiple-level association rules in large databases. *IEEE Transactions on knowledge and data engireering*, 11(7):798–805, 1999.

[53] Jiawei Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[54] V. Hatzivassiloglou, L. Gravano, and A. Maganti. An investigation of linguistic features and clustering algorithms for topical document clustering. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 224–231, 2000.

[55] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[56] A. Heuer and A. Lubinski. Data reduction - an adaptation technique for mobile environments. In *In Interactive Applications of mobile Computing (IMC apos'98)*, 1998.

[57] P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 531–538. ACM, 2008.

[58] J Hipp, A Myka, R Wirth, and U Guntzer. A new algorithm for faster mining of generalized association rules. *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD98)*, pages 74–82, 1998.

[59] T. Hofmann. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 682–687, 1999.

[60] J. Hong, E. Suh, and S. Kim. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, November 2008.

[61] E. Hovy and C.Y. Lin. Automated text summarization in SUMMARIST, 1999.

[62] Yka Huhtala, Juha Karkkainen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies, 1999.

[63] IBM. IBM Quest Synthetic Data Generation Code, 2009.

[64] D. Ienco and R. Meo. Towards the Automatic Construction of Conceptual Taxonomies. pages 327–336. Springer, 2008.

[65] Tomasz Imieliski, Leonid Khachiyan, and Amin Abdulghani. Cubegrades: Generalizing association rules. *Data Mining and Knowledge Discovery*, 6:219–257, 2002. 10.1023/A:1015417610840.

[66] Anthony Jameson. Modelling both the context and the user. *Personal Ubiquitous Comput.*, 5(1):29–33, 2001.

[67] Tianyi Jiang and Alexander Tuzhilin. Improving personalization solutions through optimal segmentation of customer bases. *IEEE Trans. Knowl. Data Eng.*, 21(3):305–320, 2009.

[68] Burn-Thornton K., Garibaldi J., and Mahdi A. Pro-active network managment using data mining. In *GLOBECOM 1998*, 1998.

[69] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The YAGO-NAGA approach to knowledge discovery. *ACM SIGMOD Record*, 37(4):41–47, 2009.

[70] J. Kivinen and H. Mannila. Approximate inference of functional dependencies from relations. *Theoretical Computer Science*, 149(1):129–149, 1995.

[71] Chan Man Kuok, Ada Fu, and Man Hon Wong. Mining fuzzy association rules in databases. *SIGMOD Record*, 27:41–46, 1998.

[72] R.Y.K. Lau, D. Song, Y. Li, T.C.H. Cheung, and J.X. Hao. Toward a fuzzy domain ontology extraction method for adaptive e-learning. *Knowledge and Data Engineering, IEEE Transactions on*, 21(6):800–813, 2009.

[73] Yeong-Chyi Lee, Tzung-Pei Hong, and Tien-Chin Wang. Multi-level fuzzy mining with multiple minimum supports. *Expert Syst. Appl.*, 34(1):459–468, 2008.

[74] Kenneth Wai-Ting Leung and Dik Lun Lee. Deriving concept-based user profiles from search engine logs. *IEEE Trans. Knowl. Data Eng.*, 22(7):969–982, 2010.

[75] Michael Ley. DBLP bibliography server, 2005. http://dblp.uni-trier.de/xml.

[76] Q. Li, J. Wang, Y.P. Chen, and Z. Lin. User comments for news recommendation in forum-based social media. *Information Sciences*, 2010.

[77] T. Li and S. Anand. *Exploiting Domain Knowledge by Automated Taxonomy Generation in Recommender Systems*. Springer, 2009.

[78] X. Li, L. Guo, and Y.E. Zhao. Tag-based social interest discovery. In *Proceeding of the 17th international conference on World Wide Web*, pages 675–684, 2008.

[79] Bing Liu, Wynne Hsu, and Yiming Ma. Mining association rules with multiple minimum supports. In *In Knowledge Discovery and Data Mining, 1999*, pages 337–341, 1999.

[80] Bing Liu, Wynne Hsu, and Yiming Ma. Discovering the set of fundamental rule changes. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 335–340, New York, NY, USA, 2001. ACM.

[81] Bing Liu, Wynne Hsu, Lai-Fun Mun, and Hing-Yan Lee. Finding interesting patterns using user expectations. *IEEE Trans. Knowl. Data Eng.*, 11(6):817–832, 1999.

[82] Bing Liu, Yiming Ma, and Ronnie Lee. Analyzing the interestingness of association rules from the temporal dimension. In *ICDM*, pages 377–384, 2001.

[83] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, U.S.A., 1967.

[84] Gunjan Mansingh, Kweku-Muata Osei-Bryson, and Han Reichgelt. Using ontologies to facilitate post-processing of association rules by domain experts. *Information Sciences*, 181(3):419 – 434, 2011.

[85] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *SIGMETRICS '05*, pages 50–60, 2005.

[86] M. Neshati and L. Hassanabadi. Taxonomy construction using compound similarity measure. In *OTM'07: Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems*, pages 915–932. Springer-Verlag, 2007.

[87] NetGroup. Analyzer 3.0, 2009.

[88] P. Nurmi, A. Salden, S. L. Lau, J. Suomela, M. Sutterer, J. Millerat, M. Martin, E. Lagerspetz, and R. Poortinga. A system for context-dependent user modeling. *OTM Federated Workshops*, 4278:1894–1903, 2006.

[89] Nuria Oliver, Ashutosh Garg, and Eric Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Comput. Vis. Image Underst.*, 96(2):163–180, 2004.

[90] Pang-Ning Tan and Michael Steinbach and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

[91] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object Exchange Across Heterogeneous Information Sources. In *ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering*, pages 251–260, Washington, DC, USA, 1995. IEEE Computer Society.

[92] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[93] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

[94] I. Pramudiono and M. Kitsuregawa. Fp-tax: tree structure based generalized association rule mining. In *DMKD '04*, pages 60–63, 2004.

[95] Python. Python website, 2009.

[96] Bridges S., Hossain M., and Vaughn Jr. R. Adaptive intrusion detection with data mining. In *IEEE International Conference on Systems, Man and Cybernetics*, 2003.

[97] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213, 1999.

[98] Bin Shen, Min Yao, Zhaohui Wu, and Yunjun Gao. Mining dynamic association rules with comments. *Knowl. Inf. Syst.*, 23:73–98, April 2010.

[99] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 259–266, 2008.

[100] O. Shi-yan, SG Khoo Christopher, and H. Goh Dion. Constructing a taxonomy to support multi-document summarization of dissertation abstracts. *Journal of Zhejiang University-Science A*, 6(11):1258–1267, 2005.

[101] B. Sigurbjornsson and R. Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceeding of the 17th international conference on World Wide Web*, pages 327–336, 2008.

[102] S Singh, P Vajirkar, and L Yugyung. Context-based data mining using ontologies. *Lecture Notes in Computer Science*, 2813(2):405–418, 2003.

[103] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB 1995*, pages 407–419, 1995.

[104] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB 1995*, 1995.

[105] K. Sriphaew and T. Theeramunkong. A new method for finding generalized frequent itemsets in association rule mining. In *Proceeding of the VII International Symposium on Computers and Communications*, pages 1040–1045, 2002.

[106] Lynda Tamine-Lechani, Mohand Boughanem, and Mariam Daoud. Evaluation of contextual information retrieval effectiveness: overview of issues and research. *Knowl. Inf. Syst.*, 24(1):1–34, 2010.

[107] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 32–41, July 2002.

[108] P.N. Tan, M. Steinbach, V. Kumar, et al. *Introduction to data mining*. Pearson Addison Wesley Boston, 2006.

[109] Yingying Tao and M. Tamer Özsu. Mining frequent itemsets in time-varying data streams. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1521–1524, New York, NY, USA, 2009. ACM.

[110] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing, Second International Conference Vienna, Austria, April*, pages 158–175, 2004.

[111] TPC-H. The TPC benchmark H. Transaction Processing Performance Council, 2009. http://www.tpc.org/tpch/default.asp.

[112] S. Tseng and C. Tsui. Mining multilevel and location-aware service patterns in mobile web environments. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(6):2480–2485, 2004.

[113] Pravin Vajirkar, Sachin Singh, and Yugyung Lee. Context-aware data mining framework for wireless medical application. In *DEXA*, pages 381–391, 2003.

[114] Keshri Verma and O. P. Vyas. Efficient calendar based temporal association rule. *SIGMOD Rec.*, 34:63–70, September 2005.

[115] World Wide Web Consortium. Web Ontology Language (OWL), 2010.

[116] Geoffrey I. Webb. Discovering significant patterns. *Machine Learning*, 71(1):131, 2008.

[117] W.L. Woon and S. Madnick. Asymmetric information distances for automated taxonomy construction. *Knowledge and Information Systems*, 21(1):91–111, 2009.

[118] XQuery. Xquery w3c website, 2009.

[119] Y. Xue, C. Zhang, C. Zhou, X. Lin, and Q. Li. An Effective News Recommendation in Social Media Based on Users' Preference. In *Education Technology and Training, 2008. and 2008 International Workshop on Geoscience and Remote Sensing. ETT and GRS 2008. International Workshop on*, volume 1, pages 627–631. IEEE, 2009.

[120] M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 31(7):1195–1209, 2009.

[121] Z. Yin, R. Li, Q. Mei, and J. Han. Exploring social tagging graph for web object classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 957–966. ACM, 2009.

[122] M Zaki, S Parthasarathy, M Ogihara, and W Li. New algorithms for fast discovery of association rules. *3rd Intl. Conf. on Knowledge Discovery and Data Mining*, 1997.

[123] O. Zamir, O. Etzioni, O. Madani, and R.M. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 287–290, 1997.

[124] Ingrid Zukerman and David W. Albrecht. Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction*, 11(1-2):5–18, 2001.